An approximate alternative to perfect simulation

Sharp, Richard Michael

Award date:
2003

Awarding institution:
University of Bath

UNIVERSITY OF
BATH

An Approximate Alternative to Perfect Simulation

submitted by

Richard Michael Sharp

for the degree of Ph.D.

of the

University of Bath

2003

An Approximate Alternative to Perfect Simulation

submitted by

Richard Michael Sharp

for the degree of Ph.D.

of the

University of Bath

2003

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Summary

Markov chain Monte Carlo techniques allow complex distributions to be investigated through simulation, however for most distributions the time taken for a chain to converge will be unknown. There are many methods available for diagnosing convergence of a chain based on its output, but in any finite run of a chain there will be uncertainty about whether it has converged.

Propp & Wilson constructed an algorithm called coupling from the past (CFTP) for removing this uncertainty and returning an exact sample from the stationary distribution of a chain. However their original algorithm can only be applied to discrete distributions with certain properties. Many new algorithms have been developed that extend CFTP to allow perfect simulation from some continuous distributions. We look at extending the perfect slice sampler of Mira, Møller & Roberts to more easily cope with unbounded distributions.

Unfortunately perfect simulation algorithms are not available for most distributions of interest. We look at using the idea of CFTP to construct an algorithm for generating an almost exact sample from a wide range of distributions, and apply it to some examples. One class of distributions for which diagnosing convergence is a problem, is distributions where the dimension of the parameter vector is not fixed. Our method does not rely on following each parameter in the model but rather diagnoses convergence of the chain as a whole, and so can be applied to such examples. We apply our method to two mixture models.

The uncertainty about whether a chain has converged can be reduced by using methods that make sure that the chain fully explores the state space. One method which finds modes of a distribution using a Metropolis-Hastings step incorporating local optimizations, and allows a chain to jump between them was introduced by Tjelmeland & Hegstad. We extend their method to allow more flexibility in the jump between modes, and apply this to the problem of finding subsets of outlying observations in Bayesian linear models.

Acknowledgements

I would like to thank my supervisor Chris Jennison for his invaluable assistance during the course of my PhD, without whose help completing this thesis would not have been possible.

David Hobson and Christian Robert should be mentioned for help with Theorem 3 and for providing C code for the mixture of regressions example respectively.

Many thanks also go to those people who survived my company in close quarters: Pete and Jeremy for far too many hours spent in the office; and Tracey, Sarah and Darran for three enjoyable years living together.

General thanks go to all those people whom I have met while at Bath, especially Mark, James and Dan, and to my parents for being around when needed.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Outline

This thesis is concerned with looking at some methods used in the field of Markov chain Monte Carlo (MCMC) techniques. In this chapter we introduce the idea behind MCMC methods particularly from a Bayesian perspective. One of the problems with using MCMC methods as a tool for investigating distributions is that of the chain's convergence. We describe the algorithm of coupling from the past (Propp & Wilson 1996) which generates an exact sample from the stationary distribution of a chain, and so overcomes this problem.

However it has proved a difficult task to extend coupling from the past (CFTP) much beyond the distributions on a finite state space for which it was first proposed and cope with general distributions arising in Bayesian inference. One possible step on the way to doing so is through the use of perfect slice sampling. In Chapter 2 we extend the perfect slice sampling algorithm of Mira, Møller & Roberts (2001) to cope better with unbounded state spaces.

Since Propp & Wilson's (1996) original paper many algorithms have been produced that extend CFTP to allow perfect simulation from continuous and unbounded state spaces, but these have proved to be difficult and impractical to sample from most distributions arising from Bayesian models. The difficulty in applying perfect simulation methods is that careful attention has to be paid to making sure the entire state space maps to the same state at time 0. In Chapter 3 we introduce a new method which will generate an almost perfect sample from a distribution, and uses the idea behind CFTP of running coupled chains from the past. This involves running a number of coupled chains from the past in such a way that there is a high chance that a coupled chain simulating from the stationary distribution will coalesce with these chains. Relaxing

the exact part of CFTP allows more flexibility in the types of chains that can be used and so we can apply our method to a much wider range of distributions.

In Chapter 4 we apply our method to a variety of distributions arising from Bayesian models. One class of distributions which present a problem for existing diagnostic methods are distributions where the parameter vector is not of fixed length, for example mixture distributions that model each observation as coming from one of a set of populations, but where the number of populations is unknown. Convergence of a chain sampling from these distributions cannot be monitored by following each parameter in the model, because the number and meaning of some of the parameters changes. Our method diagnoses convergence based on coupled chains coalescing to the same state, assessing convergence of the chains as a whole, rather than diagnosing convergence based on some summary statistics of the chains output, as most diagnostic methods do. It can therefore be applied to distributions over a varying parameter space and we apply it to a couple of mixture models. Despite being conceptually nice, some perfect simulation algorithms perform poorly in practice, and at the end of Chapter 4 we apply our method to the antiferromagnetic Potts model and show that it can obtain almost perfect samples at temperatures below that of a perfect simulation algorithm.

Tjelmeland & Hegstad (2001) introduced a method for enabling a chain simulating from a multimodal distribution to locate and then jump between widely separated modes. In Chapter 5 we extend their method to increase its flexibility and make it more convenient to use. We then look at applying the extended mode jumping algorithm to the problem of identifying outliers in Bayesian linear models, which can lead to multimodal distributions because of the presence of sets of outlying observations which mask one another.

1.2 A Problem in Statistical Inference

The problem of evaluating high dimensional integrals often arises in statistics, especially in a Bayesian context. For a random variable X with distribution proportional to $\pi(\cdot)$ the expectation of a function $f(X)$,

$$\mathbb{E}[f(X)] = \frac{\int f(x)\pi(x)dx}{\int \pi(x)dx}\,, \tag{1.1}$$

is often of interest. Such expectations often arise in Bayesian inference. Let Y denote the observed data and θ denote the model parameters. The object of all Bayesian inference is then the posterior distribution $p_{\theta|Y}(\theta|Y)$ which is constructed from the

prior $p_\theta(\theta)$ and the likelihood $p_{Y|\theta}(Y|\theta)$ using Bayes theorem,

$$p_{\theta|Y}(\theta|Y) = \frac{p_\theta(\theta)p_{Y|\theta}(Y|\theta)}{\int p_\theta(\theta')p_{Y|\theta}(Y|\theta')d\theta'} \,. \tag{1.2}$$

Any features of the posterior distribution, $p_{\theta|Y}(\theta|Y)$, can be expressed in terms of posterior expectations of functions of θ, such as moments, quantiles, and posterior probabilities of regions, however in most cases the normalisation constant, $\int p_\theta(\theta')p_{Y|\theta}(Y|\theta')d\theta'$, is unknown and so evaluating expressions of the form (1.2) is usually analytically intractable.

Simulation can be used to approximate $\mathbb{E}[f(X)]$. Monte Carlo simulation draws independent samples $\{X_t : t = 1, \ldots, n\}$ from $\pi(\cdot)$ and uses the approximation

$$\mathbb{E}[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(X_t) \tag{1.3}$$

which holds by the strong law of large numbers. The $\{X_t\}$ need not be independent as long as they have the correct distribution and so a Markov chain with stationary distribution π may be used to draw correlated samples from π. This is called Markov chain Monte Carlo and the idea behind it is described next. A short introduction on how to construct such a chain is given in Sections 1.3 and 1.4. Section 1.5 describes some existing methods for deciding when a chain is returning samples from its stationary distribution.

1.3 Markov Chain Monte Carlo (MCMC)

A Markov chain $\{X_t\}_{t=0}^{\infty}$ on $\mathcal{S}$ has stationary distribution $\pi(\cdot)$ if its transition kernel $\mathbb{P}(x, A)$ satisfies general balance

$$\int_{x\in\mathcal{S}} \pi(dx)\mathbb{P}(x, A) = \pi(A) \,. \tag{1.4}$$

If $\mathcal{F}(\mathcal{S})$ denotes the Borel σ-algebra on S, then the transition kernel $\mathbb{P}$ is a map $\mathbb{P} : \mathcal{S} \times \mathcal{F}(\mathcal{S}) \to [0, 1]$ such that $\mathbb{P}(x, A) = \mathbb{P}(X_{t+1} \in A|X_t = x)$. Nummelin (1984) proved the following theorem which states the conditions under which a Markov chain with transition kernel $\mathbb{P}$ has π as its unique invariant distribution.

Theorem 1 *Suppose $\mathbb{P}$ is π-irreducible and (1.4) holds. Then $\mathbb{P}$ is positive recurrent and has unique invariant distribution π. If $\mathbb{P}$ is also aperiodic then for π almost all*

$x \in \mathcal{S}$

$$\|\mathbb{P}^t(x, \cdot) - \pi\| \to 0 \text{ as } t \to \infty$$

where $\|\cdot\|$ *is the total variation distance*[1].

This means that if a chain is π-irreducible, aperiodic and positive recurrent then the limiting distribution of the chain is the invariant distribution regardless of the starting value of the chain.

In order to use the estimate in (1.3) a Markov chain with transition kernel $\mathbb{P}$ satisfying the conditions in Theorem 1 (including aperiodicity) needs to be found. Checking directly that a transition kernel satisfies general balance is usually very hard, however if a transition kernel satisfies the stronger condition of detailed balance, that is

$$\int_A \pi(x)\mathbb{P}(x, B)dx = \int_B \pi(x)\mathbb{P}(x, A)dx \quad \forall A, B \subseteq \mathcal{S} ,$$

then the transition kernel also satisfies general balance. Detailed balance is much easier to check directly and it is easily seen that the Metropolis-Hastings algorithm (Metropolis, Rosenbluth, Rosenbluth, Teller & Teller 1953, Hastings 1970) described below produces a Markov chain satisfying detailed balance with respect to the desired stationary distribution.

1.4 The Metropolis-Hastings Algorithm

Metropolis et al. (1953) suggested a way to build a transition kernel which satisfies detailed balance with respect to a specified stationary distribution which was later extended by Hastings (1970). For a distribution, π on $\mathcal{S}$, known only up to a normalising constant they produce a transition kernel for a Markov chain with stationary distribution π as follows.

Suppose we have a proposal density $q(\cdot|x)$ which at each iteration of the Markov chain returns a potential new state $x' \in \mathcal{S}$ given that the current state is x. This new state is then accepted as the next state of the Markov chain with probability

$$\alpha(x'|x) = \min\left\{1, \frac{\pi(x')q(x|x')}{\pi(x)q(x'|x)}\right\}, \tag{1.5}$$

otherwise the chain remains in the same state x. The transition kernel defined in this manner satisfies detailed balance. There is considerable freedom in choosing $q(\cdot|\cdot)$ it only has to ensure that the transition kernel is π-irreducible and aperiodic.

[1]The total variation distance between the distributions of two random variables X and Y on the same state space S is defined to be $\sup_B |\mathbb{P}(X \in B) - \mathbb{P}(Y \in B)|$.

Metropolis et al. (1953) originally proposed this algorithm as a single component sampler which is how it is usually used. If X can be split into components $(X^{(1)}, \ldots, X^{(n)})$ then each of these can be updated one at a time. Let the current state at time t of the Markov chain be $x_t = (x_t^{(1)}, \ldots, x_t^{(n)})$, and consider cycling through components 1 to n updating them one at a time, at every iteration. If i is the currently chosen component a candidate state $y^{(i)}$ is generated from a proposal $q_i(\cdot|x_t^{(-i)}, x_t^{(i)})$ where

$$x_t^{(-i)} = (x_{t+1}^{(1)}, \ldots, x_{t+1}^{(i-1)}, x_t^{(i+1)}, \ldots, x_t^{(n)}).$$

The candidate is then accepted as $x_{t+1}^{(i)}$ with probability

$$\alpha_i(y^{(i)}|x_t^{(-i)}, x_t^{(i)}) = \min\left\{1\ ,\ \frac{\pi(y^{(i)}|x_t^{(-i)})q(x_t^{(i)}|x_t^{(-i)}, y^{(i)})}{\pi(x_t^{(i)}|x_t^{(-i)})q(y^{(i)}|x_t^{(-i)}, x_t^{(i)})}\right\}$$

where $\pi(\cdot|x^{(-i)})$ is the full conditional distribution of the i^{th} component, otherwise $x_{t+1}^{(i)} = x_t^{(i)}$. The Markov chain defined in this way does not satisfy detailed balance, however it does satisfy general balance because the transition kernel for each component satisfies detailed balance. The order in which the components are updated need not be fixed as 1 to n, they can be updated in any predetermined order at each iteration or the order can be chosen at random. The Metropolis-Hastings algorithm is the basis for many other MCMC algorithms, some of which will be used later in this thesis and are now mentioned.

One special case of the single-component Metropolis-Hastings algorithm is the Gibbs sampler (Geman & Geman 1984) which takes

$$q_i(\cdot|x_t^{(-i)}, x_t^{(i)}) = \pi(\cdot|x_t^{(-i)})$$

resulting in an acceptance probability of 1. Some other special cases of the Metropolis-Hastings algorithm include the random walk Metropolis where $q(y|x) = q'(|y - x|)$ for some density q', and the independence sampler (Tierney 1994) which uses $q(y|x) = h(y)$ for some density h.

1.5 How Long to Run a Chain?

In order to use the approximation in (1.3), as well as producing a suitable $\mathbb{P}$ we also need to decide how long to run our Markov chain and perhaps also how many chains to run. The stationary distribution of each chain will be the same regardless of the starting state. However, it will take some iterations before the chain has "forgotten"

its starting value. This can be allowed for by having an initial burn-in where the first m iterations are not used in (1.3), and m is chosen so that thereafter the chain can be considered as generating correlated samples from π. Many schemes for doing this have been proposed and Cowles & Carlin (1996) and Brooks & Roberts (1998) review all of the more commonly used methods, so we refer the reader to these for detailed descriptions of any method used here.

The main approach to this problem has been to first decide upon a suitable Markov chain for which it is believed moves freely around the state space at an acceptable rate. One or more copies of such a chain (not necessarily independent) is then run for an initial number of iterations. The output from these chains is then analysed to see if the chain has passed its initial transient phase which is discarded. These methods are called convergence diagnostics. In this thesis we will look at the four methods popularised by the CODA (Best, Cowles & Vines 1995) suite of Splus functions, which are diagnostics based on the output of a Markov chain due to Gelman & Rubin (1992), Raftery & Lewis (1992*a*), Geweke (1992) and Heidelberger & Welch (1983), a brief description of each now follows. In this thesis the package called Bayesian Output Analysis Program (Smith 2001), BOA, was used to produce the diagnostic results, which performs the same routines as CODA but runs in R.

Gelman & Rubin (1992) propose running multiple chains which are started from over-dispersed starting values. Then for each univariate function of the chains state which is of interest, they suggest comparing the ratio of the between and within chain variances. The idea is that if the chains have converged then they should be visiting the same parts of the state space and so this ratio called the potential scale reduction factor (PSRF) should be near 1, (although to take into account the uncertainty in the estimates of variances used in the calculation some multiplicative factors are included in calculating the PSRF). Brooks & Gelman (1998) later give a corrected scale reduction factor (CSRF) to take into account extra sampling variability unaccounted for by Gelman & Rubin (1992). Gelman & Rubin (1992) recommend running 10 chains if there is one mode and more for a multimodal distribution, although it has become common to use around 5 chains.

Raftery & Lewis (1992*a*) suggest running one chain and for each univariate function of interest of the chain obtaining a burn-in length based on estimating specified quantiles of the stationary distribution. Their method is based on two-state Markov chain theory for a chain moving between the two states of being in the specified quantile of the distribution or not, and whether its n-step transition probabilities have got close enough to its stationary distribution. Formulas are given for the recommended burn-in lengths and also how long thereafter the chain should be run to get estimates

of confidence intervals for the specified quantiles to a given degree of accuracy.

Geweke (1992) describes a method for diagnosing convergence for one chain which produces a statistic based on two means, one calculated using a proportion of states from the beginning of the run and the other using a proportion from the end of the run. The statistic (the difference of the means divided by the sum of their estimated variances) should converge to the standard Normal distribution if the chain is stationary. If the values of this statistic do not look to have come from a standard Normal distribution then it should be concluded that the chain has not converged. However Geweke (1992) does not give any further guidance on applying this diagnostic.

Heidelberger & Welch (1983) give a diagnostic that says whether a chain has reached stationarity or not in the first half of the available iterations. A test is also made of whether the mean is sufficiently accurately estimated from the remaining samples after the burn-in has been discarded.

Of course one diagnostic alone should not be relied upon. Several diagnostics should be used as well as inspection of trace plots of each parameter, mode finding algorithms and any other methods to get a better understanding a chains behaviour, see for example comments in Cowles & Carlin (1996), Brooks & Roberts (1998) and Brooks & Gelman (1998).

When trying to assess the convergence of a chain it has to be decided how many chains to run, and this has been an often discussed topic. The main argument is whether to run one long chain or several shorter ones, given a fixed amount of computation time. It has been argued that running one long chain is better (Raftery & Lewis 1992*b*) because the samples from the end of a long run will always be closer to the stationary distribution than those from the shorter runs. However with a Markov chain that is slow to move through the state space, for example when the stationary distribution has well separated modes, then one chain may not reach all of the modes, which is the argument frequently given by those arguing for multiple chains. The problem of chains not navigating the whole space is not confined to multimodal distributions, another example of a slowly mixing chain is one simulating from a heavy tailed distribution which uses a Metropolis-Hastings proposal with light tails, which can be slow to move to and from the tails of its stationary distribution. Some of the uncertainty in the mixing of the chain can be resolved by running multiple chains with starting values chosen from a distribution that is over-dispersed relative to the stationary distribution. The over-dispersed distribution should have significant probability in each of the areas of the stationary distribution that the chain has trouble moving between. Gelman & Rubin (1992) describe a method for finding such an over-dispersed distribution, based on locating the modes of the stationary distribution and then approximating each

mode with a t-distribution. The starting states are then obtained using importance resampling from the mixture of t-distributions.

It is however inherent in methods that look at a finite simulation of a chain that it cannot be guaranteed that the chain has converged. The next section describes a method due to Propp & Wilson (1996) which overcomes this problem by returning a sample from the stationary distribution of a chain in finite time and this removes the need for a burn-in period.

1.6 Perfect Simulation

Propp & Wilson (1996) describe a method called coupling from the past (CFTP) that returns a sample from the stationary distribution π of a Markov chain on a finite space Ω by running coupled chains starting from every state, from some time in the past up to time 0, so that they all coalesce to the same state. This method is impractical for large state spaces, but if the state space has a partial ordering which is preserved by the transition matrix Propp & Wilson (1996) show that only two chains need to be followed.

A convenient way to describe a Markov chain especially when discussing CFTP is by the use of a stochastic recursive sequence (SRS) which is equivalent to the Markov chain. As the SRS and Markov chain are describing the same process I will use the notation $\{x_t\}_{t=s}^{\infty}$ interchangeably to denote the states of both. The SRS equivalent to a Markov chain can be written in terms of a doubly infinite sequence of independent identically distributed random variables $\{\gamma_t\}_{t=-\infty}^{\infty}$ and an update function $\Psi : \Omega \times \mathcal{R} \to \Omega$, where $\mathcal{R}$ denotes the space of possible values of each γ_t. The initial state of the Markov chain and SRS is the same, x_s, and

$$x_{t+1} = \Psi(x_t, \gamma_t)$$

updates the SRS according to the transition probabilities specified by the Markov chain. We note that although for convenience a single doubly infinite sequence of random variables has been used, this is not necessary. The SRS could use more independent sequences or each γ_t itself could be a sequence of independently identically distributed random variables. We now describe the CFTP algorithm and give a proof of its correctness.

Define $x_t^{(x,T)}$ to be the value at time t, $(t = T, T+1, T+2, \ldots)$, of a chain started at time T in state $x \in \Omega$, and for notational convenience we let $\Omega = \{1, \ldots, k\}$. In order that CFTP returns a sample from π, Ψ must be chosen so that all chains are guaranteed to eventually coalesce. CFTP proceeds by first choosing a time $T_1 < 0$ and

producing k stochastic recursive sequences with

$$x_{T_1}^{(1,T_1)} = 1, \ldots, x_{T_1}^{(k,T_1)} = k$$

and

$$x_{t+1}^{(1,T_1)} = \Psi(x_t^{(1,T_1)}, \gamma_t), \ldots, x_{t+1}^{(k,T_1)} = \Psi(x_t^{(k,T_1)}, \gamma_t)$$

for $t = T_1, T_1 + 1, \ldots, -1$. Then if

$$x_0^{(1,T_1)} = x_0^{(2,T_1)} = \cdots = x_0^{(k,T_1)}$$

this value is returned as a sample from π, however if the chains have not coalesced at time 0 then we choose a new time $T_2 < T_1$ and restart the stochastic recursive sequences from time T_2 making sure to use the same γ_t from time T_1 to time 0. We now give a proof of the correctness of this algorithm, which is a shortened version of the proof given by Casella, Lavine & Robert (2001).

Theorem 2 *Let $X_t^{(x,T)}$ denote the random state of a Markov chain at time t started in state $x \in \Omega$ at time T. Let this chain be equivalent to a SRS using an update function Ψ. If this chain satisfies the conditions of Theorem 1 and also has the property that two coupled chains updated using Ψ will eventually coalesce, that is for all $x, y \in \Omega$ there exists an $N < 0$ such that*

$$\mathbb{P}(X_0^{(x,N)} = X_0^{(y,N)}) > 0,$$

then with probability 1 the procedure described above returns a value and this value is distributed according to π.

Proof

Let $p(X)$ denote the distribution of a random variable X. For a sequence of random variables $\{Y_t\}_{t=0}^{\infty}$ let $p(Y_t) \to g$ denote the convergence in probability of the distribution of the Y_t to g as $t \to \infty$. If we can show that the following three statements are true then the theorem holds. The notation $X_t^{(x,T)}$ for random variables follows the same convention as for observed values $x_t^{(x,T)}$.

1. The algorithm finishes in finite time and produces a value, X_0.
2. For each $j \in \Omega$, $p(X_0^{(j,-t)}) \to \pi$ as $t \to \infty$.
3. For each $j \in \Omega$, $X_0^{(j,-t)} \to X_0$ as $t \to \infty$.

We will then have $X_0 \sim \pi$.

1. The guaranteed coalescence condition which Ψ must satisfy ensures that for each $j \in \Omega$ there exists an N_j such that

$$\mathbb{P}(X_0^{(j,N_j)} = x) > 0 \quad \forall x \in \Omega.$$

Setting $N = \min\{N_1, \ldots, N_k\}$ it follows that for some $\epsilon > 0$

$$\mathbb{P}(X_0^{(1,N)} = X_0^{(2,N)} = \cdots = X_0^{(k,N)}) > \epsilon. \tag{1.6}$$

Now let C_i be the event that

$$X_{(i-1)N}^{(1,iN)} = X_{(i-1)N}^{(2,iN)} = \cdots = X_{(i-1)N}^{(k,iN)}$$

which is the event that the k chains starting at time iN have coalesced by time $(i-1)N$. The C_i are independent as the $\{\gamma_t\}_{t=-\infty}^{\infty}$ are independent and by (1.6) we have that $\mathbb{P}(C_i) > \epsilon$ for each i. It is then the case that

$$\begin{aligned} \mathbb{P}(\text{no coalescence after } IN \text{ iterations}) &\leq \prod_{i=1}^{I}[1 - \mathbb{P}(C_i)] && (1.7) \\ &< (1-\epsilon)^I && (1.8) \\ &\to 0 \text{ as } I \to \infty, && (1.9) \end{aligned}$$

showing that the probability of coalescence is 1 and that the algorithm finishes in finite time.

2. Observe that $p(X_0^{(j,-t)}) = p(X_t^{(j,0)})$ because they are both the distribution of a Markov chain starting from state j having progressed t iterations. From Theorem 1 we have that π is the stationary distribution of the chain and so $p(X_0^{(j,t)}) \to \pi$ as $t \to \infty$.

3. By 1 there exists an N for which coalescence has occurred between time N and time 0. So for all $t \leq N$, $X_0^{(j,t)} = X_0$ which implies fact 3 is true. □

In practice for any examples for which this method would be useful there will be a large number of states, and so it will be computationally infeasible to follow chains starting in every state. However if the chain and state space exhibit certain properties then it is only necessary to follow a small number of chains in order to check coalescence of the whole state space.

Propp & Wilson (1996) originally noted the following situation under which the algorithm can be made more efficient. Suppose the state space has a partial ordering

$\preceq$ which the update function preserves, so that

$$x \preceq y \iff \Psi(x,\gamma) \preceq \Psi(y,\gamma) \qquad \forall\gamma. \tag{1.10}$$

This is called a monotone Markov chain. Suppose also that the state space has minimum and maximum elements $\hat{0}$ and $\hat{1}$ respectively such that $\hat{0} \preceq x \preceq \hat{1}$ for all $x \in \Omega$. If for some $T < 0$, $x_0^{(\hat{0},T)} = x_0^{(\hat{1},T)}$, then for all $j \in \Omega$ we have that $x_0^{(j,T)} = x_0^{(\hat{0},T)}$ as chains starting from other states are sandwiched between the minimum and maximum chains by (1.10). We now only need to follow two chains starting from the minimum and maximum elements.

In implementing the algorithm there remains the choice of what times in the past to start the algorithm. Propp & Wilson (1996) suggest first trying $T_1 = -1$ and then doubling back with $T_2 = 2T_1$ and so on until coalescence occurs. This strategy is no worse than four times the best possible outcome in which the user clairvoyantly chooses the smallest negative T, call it T^* such that $x_0^{(\hat{0},T^*)} = x_0^{(\hat{1},T^*)}$. This can be seen by observing that the doubling strategy will take less than[2] $2(1 + 2 + \cdots + 2^k)$ iterations to complete where k is such that $2^{k-1} < -T^* \leq 2^k$ and where $2^{k-1} < -T^*$ because otherwise $T = -2^k$ would not have needed to be tried. The factor of 2 comes from running both the minimum and maximum chains. We also have that

$$2(1 + 2 + \cdots + 2^k) < 2^{k+2}$$

showing that the number of iterations required, $2(1 + 2 + \cdots + 2^k)$, is no more than 4 times $-2T^*$ which is the number of iterations needed by the clairvoyant user who wishes to confirm that $x_0^{(\hat{0},T^*)} = x_0^{(\hat{1},T^*)}$ by following both chains.

This form of coupling from the past has been successfully applied to many examples from statistical physics including among others the Potts model, which will be used later as an example to demonstrate the use of our algorithm for generating almost perfect samples.

Before moving on to discuss extensions to Propp & Wilson's (1996) original CFTP algorithm we mention the existence of the "other perfect simulation algorithm" due to Fill (1997). Fill's (1997) algorithm is more complicated to implement and so has not been as extensively used, but unlike CFTP it is interruptible. Fill's (1997) algorithm, like CFTP, consists of choosing a sequence of times, and then going through each time checking whether a set of coupled chains have coalesced. Both algorithms stop when a coalescence time is found and return a sample from the stationary distribution of the chains. CFTP must use the same set of random variables to update each chain for every

[2]Once two chains coalesce only one of them needs to be followed.

choice of time otherwise it does not return a sample from the desired distribution, it therefore cannot be stopped and restarted with a different set of random variables. Fill's (1997) algorithm however does not use the same random variables when investigating each time and so may be stopped at any moment and restarted with a different sequence of random variables. It is this property of being able to restart the algorithm with a new set of random variables that defines an interruptible algorithm.

1.7 Extensions to Propp and Wilson's CFTP

Many authors have developed variations of CFTP and methods which allow CFTP to be used to sample from a larger range of distributions, for example read-once CFTP (Wilson 2000*a*), multi-stage sampling (Meng 2000), exact sampling using summary states (Huber 1998), CFTP with anti-monotone chains (Häggström & Nelander 1998), and CFTP using dominating processes (Kendall 1997, Kendall & Møller 2000). It is difficult to apply such methods in a general Bayesian setting. One reason for this is because ordinarily coupled Markov chains will not coalesce in a continuous space, they only become arbitrarily close, and all of the above algorithms rely on discrete components in the state space to enable the coalescing of chains. Murdoch and Green (Murdoch & Green 1998, Green & Murdoch 1999) have introduced several CFTP algorithms for perfect simulation in continuous spaces which reduce a potentially infinite number of chains to a finite number. However most of the algorithms require the transition kernel or stationary distribution to have certain properties, or be related to other simpler distributions, meaning that these methods are difficult to apply in a general setting. We now give short descriptions of some of the perfect simulation algorithms which are related to some of the ideas used later.

The multigamma coupler (Murdoch & Green 1998) relies on finding a non-negative function $r(\cdot)$ lying entirely under all the transition kernels $f(\cdot|\cdot)$, that has the property that

$$f(y|x) \geq r(y) \qquad \forall x, y \in \mathcal{S}$$

and $\rho = \int r(y)dy > 0$. Then a Markov chain whose state is updated using f can instead be updated using the mixture kernel

$$r(y) + q(y|x) \tag{1.11}$$

where

$$q(y|x) = f(y|x) - r(y).$$

Samples from (1.11) can be obtained by sampling from the distribution proportional to

$r(\cdot)$ with probability ρ and from the distribution proportional to $q(\cdot|\cdot)$ with probability $(1-\rho)$. On those iterations where the next state of the chain, y, is obtained using $r(\cdot)$ all chains regardless of their current state will move to y and coalesce.

The rejection coupler (Murdoch & Green 1998) can be used when $f(\cdot)$ is known only up to a normalising constant say $f(y|x) = k(x)g(y|x)$ where $k(x)$ is unknown. It requires the user to be able to find functions $r(\cdot)$ and $h(\cdot)$ satisfying

$$0 \leq r(y) \leq g(y|x) \leq h(y) \qquad \forall x, y \in \mathcal{S} \tag{1.12}$$

where

$$\nu = \int h(y)dy < \infty \,,$$

$$\rho = \int r(y)dy \;/ \int h(y)dy > 0$$

and $h(\cdot)/\nu$ can be easily sampled from. The rejection coupler can be used to provide a finite list of potential next states for a chain regardless of its current state. At each iteration rejection sampling is used to obtain a sample from the density proportional to $r(\cdot)$ using $h(\cdot)$ as the envelope. Let $(y_1, \xi_1), \ldots, (y_k, \xi_k)$ be the sequence of samples from under $h(\cdot)$ in the order they were generated, so $\xi_k \leq r(y_k)$. Any individual chain currently in state z say is also updated using rejection sampling with envelope $h(\cdot)$, so its next state will be $z' = y_{j'}$ where $j' = \min\{j : \xi_j \leq g(y_j|z)\}$ which is the first sample lying under $g(\cdot|z)$. The fact that $r(y) \leq g(y|x)$ for all x means that there is at least one such y_j in the set $\{y_1, \ldots, y_k\}$. For the purposes of CFTP we now only need follow chains starting from the finite list of potential next states $y_1, \ldots, y_k$. Repeating this procedure using only the remaining finite list of possible states the number of chains will eventually be reduced to one.

The Metropolis-Hastings coupler (Murdoch & Green 1998) relies on finding an over-dispersed independence proposal, defined to be a density $q(\cdot)$ such that

$$\sup \frac{\pi(x)}{q(x)} = K < \infty. \tag{1.13}$$

Then if y is the proposed next state sampled from $q(\cdot)$, the usual acceptance probability (1.5) satisfies

$$\alpha(y|x) \geq \frac{\pi(y)}{q(y)K}.$$

When coupling chains, at each iteration the same random variable U from a $U(0,1)$

distribution will be used by every chain in the rejection step, and so if

$$U \leq \frac{\pi(y)}{q(y)K}$$

all chains regardless of their current state will accept y as their next state. All chains then having coalesced. Corcoran & Tweedie (1998) have also used an independence Metropolis-Hastings chain to construct a perfect simulation algorithm that relies on finding an over dispersed proposal. The form of π will generally only be known up to a normalising constant, so that $\pi(x) = kg(x)$ where $g(\cdot)$ is known but k is an unknown constant. By ordering the state space according to the ratio $g(x)/q(x)$ Corcoran & Tweedie (1998) show how to construct a monotone Markov chain.

To make these algorithms more efficient Murdoch & Green (1998) suggest partitioning the state space and using different envelopes or proposals in each element of the partition. The idea of partitioning the state space and then attempting to get chains whose current states are in each element of the partition to coalesce is used later on. In the next section we extend the perfect slice sampler, which is another perfect simulation algorithm that coalesces chains in a continuous space.

Chapter 2

Extending Perfect Slice Sampling

2.1 Introduction

Mira et al. (2001) describe a perfect simulation algorithm based on the simple slice sampler, where a single auxiliary variable is introduced that acts to slice the target density horizontally. We show how the same method of perfect simulation can be applied to a more general class of slice samplers in a way that overcomes some of the problems with the version given by Mira et al. (2001). This extends the class of distributions that may be sampled from exactly and provides possible speed improvements. We then apply our new sampler to some truncated Gamma densities and a simple Bayesian example.

2.2 The Simple Slice Sampler

Suppose that we wish to generate samples from the distribution proportional to $f_X(x)$, $x \in \mathcal{X} \subseteq \mathbb{R}^d$ (generally the distribution of interest will only be known up to a normalizing constant). The simple slice sampler is a Gibbs sampling scheme that simulates from an extended state space via the introduction of an auxiliary variable, where the marginal distribution of the X variable is proportional to $f_X(x)$. A latent variable $U \in (0, \infty)$ is introduced whose joint density with X is given by,

$$f_{XU}(x, u) \propto I[u < f_X(x)]$$

where $I[\cdot]$ denotes the indicator function. The marginal distribution for X is proportional to $f_X(x)$ and the conditional distributions are

$$(X|U = u) \sim U(\{x : u < f_X(x)\}) \tag{2.1}$$

$$(U|X = x) \sim U(0, f_X(x)) \tag{2.2}$$

where $U(A)$ denotes a Uniform density over the set $A \subseteq \mathcal{X}$ and $U(a,b)$ a Uniform density on the interval (a,b). The simple slice sampler is defined to be the Gibbs sampler that successively simulates values of X and U using the conditional distributions (2.1) and (2.2). It can be used to simulate from $f_{XU}(x,u)$ and then the sampled values of X can be considered as samples from $f_X(x)$.

The simple slice sampler has appealing properties, for example Mira & Tierney (2002) show that if $f_X(x)$ is bounded then the simple slice sampler is uniformly ergodic which should lead to quick convergence to the stationary distribution. It is also observed by Roberts & Rosenthal (1999*a*) that the simple slice sampler (first updating U and then X) is stochastically monotone with respect to the partial ordering defined by,

$$(x,u) \preceq (x',u') \iff f_X(x) \leq f_X(x'), \tag{2.3}$$

where a Markov chain Y on a space with partial ordering $\preceq$ is said to be stochastically monotone if, when $x_1 \preceq x_2$ for every z $\mathbb{P}(Y_1 \leq z | Y_0 = x_1) \geq \mathbb{P}(Y_1 \leq z | Y_0 = x_2)$.

2.3 Perfect Slice Sampling

Mira et al. (2001) observed that if minimum and maximum elements exist in the partial ordering, and if a scheme can be found that updates a Markov chain simulating from $f_{XU}(x,u)$ that preserves the partial ordering (a monotone Markov chain) and has the possibility of two chains coalescing, then Propp & Wilson's (1996) CFTP algorithm can be used to obtain perfect samples from $f_{XU}(x,u)$. The X component of the sample returned will then be a sample from $f_X(x)$.

To describe perfect slice sampling algorithms we will use the SRS notation from Section 1.6, with Ψ for the transition function and $\theta = (x,u)$ denoting the state of a SRS. Following previous notation we have $\theta_t^{(\theta,T)}$ as the value of the SRS at time t having started in state θ at time $T \leq t$. For the perfect slice samplers we take each γ_t to be a sequence of independent $U(0,1)$ random variables, $\gamma_t = \{\gamma_t^i\}_{i=-\infty}^0$.

Let the minimum and maximum elements of the state space according to the partial ordering $\preceq$ defined in (2.3) be $\theta_{\min} = (x_{\min}, \cdot)$ and $\theta_{\max} = (x_{\max}, \cdot)$ respectively, which do not depend on the value of u as the U component is updated first. For perfect slice sampling to work Mira et al. (2001) produce a transition function that preserves the partial ordering, updating the Markov chain in accordance with the correct transition kernel, and exhibits the possibility of two chains from a continuous state space coalescing. If a $T < 0$ can be found such that

$$\theta_0^{(\theta_{\min},T)} = \theta_0^{(\theta_{\max},T)},$$

then $\theta_0^{(\theta_{\min},T)}$ is a sample from $f_{XU}(x,u)$ and the x component of this is a sample from $f_X(x)$.

We now describe the idea behind how a particular chain gets updated and how it has a chance of coalescing with chains starting from different states. Firstly the latent variable U is updated in a straightforward manner directly from its conditional distribution given in (2.2), so for a chain in state (x,u) at time $t-1$ set the U component at time t to be

$$U_t(x) = \gamma_t^0 f_X(x).$$

Two chains with the same x value at time $t-1$ will then also have the same u value at time t which keeps two chains together once they have coalesced. To coalesce the X variable Mira et al. (2001) use a continuous version of maximal coupling (Reutter & Johnson 1995). Consider two chains in states $(x_1, U_t(x_1))$ and $(x_2, U_t(x_2))$ with $(x_1, U_t(x_1)) \preceq (x_2, U_t(x_2))$. We have that

$$\{x : U_t(x_2) < f_X(x)\} \subseteq \{x : U_t(x_1) < f_X(x)\}$$

because $U_t(x_1) \leq U_t(x_2)$. So a $U(\{x : U_t(x_1) < f_X(x)\})$ distribution can be used as a rejection envelope for the $U(\{x : U_t(x_2) < f_X(x)\})$ distribution. Both chains can be updated and have a chance of coalescence by first sampling $x^{(1)}$ from a $U(\{x : U_t(x_1) < f_X(x)\})$ distribution as the next value for the first chain, and if $U_t(x_2) < f_X(x^{(1)})$ we can accept $x^{(1)}$ as a new value for both chains. If not the next value of the second chain is sampled from a $U(\{x : U_t(x_2) < f_X(x)\})$. The whole state space can be updated in this way to a finite list of values by starting from the minimum state and repeating the rejection sampling until an update for the maximum chain has been obtained. Formally this is described as follows.

For $t > 0$ let the current state of a Markov chain be (x,u), (or the equivalent stochastic recursive sequence). In updating the value of x the following sequence of random variables, $\mathbf{w}_t = \{w_{t,j} : j = 1,2,\ldots\}$ is required. Using the notation given in Mira et al. (2001), the sequence is formed by generating

$$w_{t,1} \sim U(\{x : U_t(x_{\min}) < f_X(x)\}) = U(\mathcal{X})$$

and

$$w_{t,j} \sim U(\{x : f_X(w_{t,j-1}) < f_X(x)\}), \qquad j = 2,3,\ldots$$

obtained using the remaining uniform random variables $\gamma_t^1, \gamma_t^2, \ldots$. Now define $\sigma_t(x)$ to be

$$\sigma_t(x) = \inf\{j \geq 1 : f_X(w_{t,j}) > U_t(x)\}.$$

The update function Ψ is now defined as

$$\Psi((x,u),\gamma_t) = (w_{t,\sigma_t(x)}, U_t(x)).$$

It is clear that $\sigma_t(x)$ is finite for all x because with probability 1 eventually a value $w_{t,j}$ will be obtained with $f_X(w_{t,j}) > U_t(x_{\max})$ and we will have generated the next state for every possible chain. It is the case that Ψ is monotonic as the sequence $w_{t,j}$ is increasing with respect to the ordering $\preceq$. The fact that this Markov chain is simulating from a simple slice sampler can be seen by observing that the above procedure defines an adaptive rejection sampling scheme.

The perfect slice sampling algorithm can now be given as,

1. Choose times $T_1, T_2, \ldots$ such that $0 > T_1 > T_2 > \ldots$.
2. Set $i = 1$.
3. Calculate $\theta_0^{(\theta_{\max}, T_i)}$ and $\theta_0^{(\theta_{\min}, T_i)}$ using the transition function Ψ, and then if $\theta_0^{(\theta_{\max}, T_i)} = \theta_0^{(\theta_{\min}, T_i)}$ we are done, and can accept $\theta_0^{(\theta_{\max}, T_i)}$ as a sample from $f_{XU}(x,u)$.
4. Otherwise increase i by 1 and restart step 3.

Mira et al. (2001) note that the values of $x_{\min}$ and $x_{\max}$ do not need to be attainable. If a minimum state does not exist, that is there does not exist an $x_{\min}$ such that $f_X(x_{\min}) = \inf f_X(x)$, then the minimum chain can be started in a state with X component

$$x_{\min} \sim U(\mathcal{X}).$$

However this requires the ability to sample from the uniform distribution on the whole state space which cannot be done with an unbounded state space. If the maximum state is not attainable, that is there does not exist an $x_{\max}$ such that $f_X(x_{\max}) = \sup f_X(x)$, then the maximum chain can be started in a state with X component

$$x_{\max} = w_{T_i, \bar{\sigma}_{T_i}}$$

where

$$\bar{\sigma}_t = \inf\{j \geq 1 : f_X(W_{t,j}) \geq \gamma_t^0 \sup f_X(x)\}.$$

However this still requires that $f_X(x)$ is bounded above which will limit the number of situations for which this perfect simulation method is available. Mira et al. (2001) use dominating processes to overcome these problems, but it can be hard (if not impossible) to find suitable dominating processes and using them slows the algorithm down. In the

next section we show how in some situations both of these problems may be overcome by using a different choice for the joint distribution of X and U, resulting in a new algorithm for perfect simulation.

2.4 Extending the Perfect Slice Sampler

Edwards & Sokal (1988) and Besag & Green (1993) introduced a more general slice sampler into the statistical literature. They showed how an auxiliary variable can be defined by first splitting a density into the product of two terms, e.g.

$$f_X(x) \propto \pi(x)l(x) \tag{2.4}$$

where $\pi(x)$ is chosen to be proportional to a density for which it is relatively easy to sample from it truncated to subsets of $\mathbb{R}^d$. Following the same method as used in Section 2.3 a slice sampler can now be defined by introducing an auxiliary variable $U \in (0, \infty)$ that has a joint density with X of

$$f_{XU}(x,u) \propto \pi(x)I[u < l(x)],$$

and leads to conditional distributions

$$f_{X|U}(x|u) \propto \pi(x)I[u < l(x)] \tag{2.5}$$

and

$$(U|X = x) \sim U(0, l(x)).$$

This is what Roberts & Rosenthal (1999*b*) call a polar slice sampler.

We now show how when $l(\cdot)$ is bounded above we can construct a perfect polar slice sampler. This is done in a similar manner to the one given by Mira et al. (2001) and described in Section 2.3. As noted in Mira & Tierney (2002) the slice sampler which we will define in this manner is uniformly ergodic and so should have good convergence properties. We redefine the partial ordering $\preceq$ to be

$$(x_1, u_1) \preceq (x_2, u_2) \iff l(x_1) \leq l(x_2),$$

and given the same $\{\gamma_t\}_{t=-\infty}^{0}$ redefine

$$U_t(x) = \gamma_t^0 l(x).$$

At each time t, let the sequence of $w_{t,j}$'s be

$$w_{t,1} \sim p_{t,1}(x) \quad \text{where} \quad p_{t,1}(x) \propto \pi(x)$$

and

$$w_{t,j} \sim p_{t,j}(x) \quad \text{where} \quad p_{t,j}(x) \propto \pi(x) I[l(w_{t,j-1}) < l(x)] \qquad j = 1, 2, \ldots.$$

As in Section 2.3 define $\sigma_t(x) = \inf\{j \geq 1 : l(w_{t,j}) > U_t(x)\}$ and the update function as $\Psi((x,u),\gamma_t) = (w_{t,\sigma_t(x)}, U_t(x))$. For the same reasons as given for the simple slice sampler, $\Psi(\cdot,\cdot)$ is monotone and performs updates from the polar slice sampler.

CFTP can now be done in the same way as in the previous section, but now at time $-T$ the maximum chain is started in state $(x_{\max},\cdot)$ where $l(x_{\max}) = \sup l(x)$, and the minimum chain is started in state $(x_{\min},\cdot)$ where $x_{\min} = w_{-T,1}$. In the same manner as in Section 2.3, the standard CFTP algorithm can now be used to obtain a sample from $f_{XU}(x,u)$ using our new update function and starting the minimum and maximum chains in the given states.

We now show how our extended perfect slice sampler overcomes the problems with Mira et al.'s (2001) perfect simple slice sampler noted at the end of Section 2.3. Either as a starting state for the minimum chain or the first time the minimum chain is updated, both perfect slice samplers require a sample from the limiting distribution

$$\lim_{u \to 0} f_{X|U}(x|u).$$

For Mira et al.'s (2001) sampler this is a Uniform distribution on the whole state space. If the state space is unbounded then this distribution does not exist and Mira et al. (2001) have to find a dominating process to allow perfect simulation. However for our perfect slice sampler we have

$$\lim_{u \to 0} f_{X|U}(x|u) \propto \pi(x),$$

which (by the choice of $\pi(\cdot)$) is proportional to a density on the whole state space and so unbounded state spaces present no problem.

The other problem which Mira et al.'s (2001) algorithm has is that when $f_X(x)$ is unbounded above, for any time t, the sequence of $\{w_{t,j} : j = 1, 2, \ldots\}$ does not terminate which is required for perfect simulation. So again Mira et al. (2001) have to find a dominating process in these circumstances. Our perfect slice sampler can potentially overcome the problem of an unbounded $f_X(x)$ by choosing $l(\cdot)$ to be bounded above and $\pi(\cdot)$ to contain the unbounded part of $f_X(\cdot)$. For each time t the new sequence

of $\{w_{t,j} : j = 1, 2, \dots\}$ is then finite and the state space is reduced to a finite set of possible values that a chain could take, allowing perfect simulation.

As in perfect simple slice sampling $x_{\max}$ need not be attainable because the maximum chain may be started with

$$x_{\max} = w_{-T,\bar{\sigma}_{T_i}}$$

where

$$\bar{\sigma}_t = \inf\{j \geq 1 : l(w_{t,j}) > \gamma_t^0 \sup l(x)\}. \tag{2.6}$$

In fact it is not necessary to know the exact value of $\sup l(x)$. If $\sup l(x)$ is difficult or impossible to obtain, for example if it is found using a numerical optimisation routine, but values U_A and U_B can be found where $U_A \geq \sup l(x)$ and U_B is chosen so that it is known that there exists a y such that $l(y) \geq U_B$, with U_A and U_B preferably being close to $\sup l(x)$, then perfect polar slice sampling can proceed. In (2.6) we can replace $\sup l(x)$ with U_A and if it happens that at a time T_i when we restart the chains $\gamma_{T_i}^0 U_A > U_B$ then we simply have to restart the chains from time T_{i+1}. Here we are simply looking for a $\gamma_{T_{i+1}}^0$ such that $\gamma_{T_{i+1}}^0 U_A \leq U_B$ and so because the sequence of γ_i^0's are independent we need only choose $T_{i+1} = T_i - 1$. Only at those times when the chains are restarted is it necessary to do this because after the maximum chain is initialised then we only need to follow it and do not need to know the value of $\sup l(x)$ to do so. Let $\hat{x}_t$ be the state of the X variable of the maximum chain at time t. At each iteration the generation of the $w_{t,j}$ stops when a value y is sampled from the density proportional to $\pi(\cdot)$ with

$$U_t(\hat{x}_t) < l(y)$$

and because with probability 1 $\hat{x}_t \neq \sup l(x)$ then the value of $\sup l(x)$ is not needed to determine this.

In fact this strategy of pushing CFTP a little further into the past can be used at any time the algorithm is being slow or cumbersome. For example if when restarting the minimum and maximum chains at a time T_i the value of $\gamma_{T_i}^0$ is large, then to avoid the subsequent generation of lots of $w_{t,j}$'s we can go back further in time to T_{i+1} until we find a suitably smaller $\gamma_{T_{i+1}}^0$.

2.5 A Method for Increasing the Speed of the Perfect Slice Sampler

2.5.1 Introduction

In this section we look at a possible way to reduce the time taken by the perfect slice sampler, both the sampler used by Mira et al. (2001) and the perfect polar slice sampler from Section 2.4. It may also be possible to use the method described here to improve the efficiency of other perfect simulation algorithms. Two properties of the update function which a lot of perfect simulation algorithms for sampling from continuous state spaces require are, monotonicity, and the ability to coalesce chains. We consider separating these two properties by using two different transition functions. This could result in an improved perfect simulation algorithm because a single transition function would have to satisfy both of these requirements and so may not be as efficient.

2.5.2 A General Algorithm

Here we look at improving the perfect polar slice sampler which is a generalisation of Mira et al.'s (2001) algorithm. We assume that we are able to implement a transition function Ψ which maps all states close enough in the partial ordering to the same state, but that this is computationally expensive and that we would like to minimise the number of times it is used. This could be the transition function described earlier in Section 2.4 but where sampling from (2.5) is difficult and slow. Other choices are possible, Ψ need not be a perfect sampling scheme itself, and these are discussed later.

For convenience we drop the auxiliary variable u which can be viewed as a construction in updating the state x. For the purposes of this algorithm we define $x_{\min}$ such that $l(x_{\min}) = 0$ and $x_{\max}$ such that $l(x_{\max}) = \sup l(x)$. If either $x_{\min}$ or $x_{\max}$ are not in $\mathcal{X}$ then we consider a perfect sampler defined on the extended state space

$$\bar{\mathcal{X}} = \mathcal{X} \cup \{x_{\min}, x_{\max}\},$$

which will obtain a perfect sample from the distribution

$$\bar{f}_X(x) = \begin{cases} f_X(x) & \text{if } x \in \mathcal{X} \\ 0 & \text{if } x = x_{\min}, x_{\max}. \end{cases}$$

We now suppose that we can construct another transition function Φ such that, given a sequence of $U(0,1)$ random variables γ, the next state of the Markov chain is $\Phi(x, \gamma)$,

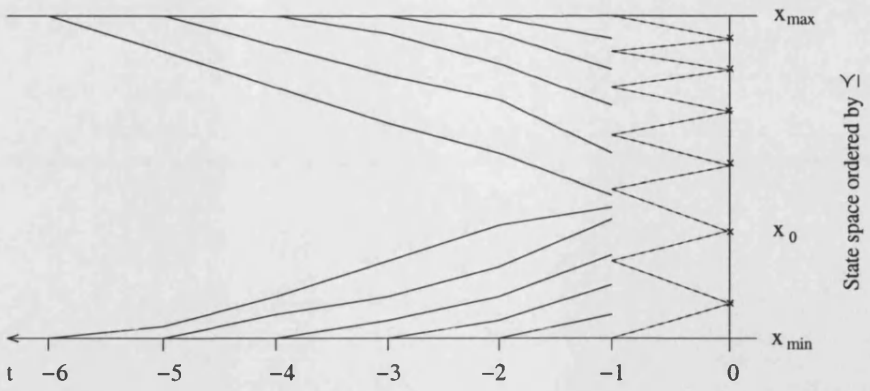

Figure 2.1: Diagram to show how to obtain a perfect sample (x_0) using Ψ and Φ. The dashed lines indicate which parts of the state space go to which of the possible updates using Ψ, and the solid lines indicate the paths of the minimum and maximum chains starting from further and further in the past updating using Φ. Eventually we go back far enough in the past that the minimum and maximum chains both accept the same state at time 0.

which has the following properties,

$$x \preceq y \iff \Phi(x, \gamma) \preceq \Phi(y, \gamma),$$

$$\Phi(x_{\min}, \gamma) \in \mathcal{X},$$

$$\Phi(x_{\max}, \gamma) \in \mathcal{X},$$

and

$$\Phi(x_{\min}, \gamma) \preceq \Phi(x_{\max}, \gamma).$$

It is not necessary for Φ to be able to coalesce chains in a continuous space, otherwise we could use Ψ instead of Φ. These conditions amount to requiring that Φ is monotone and that we can follow the minimum and maximum chains. A perfect sampler can now be constructed by using Ψ only at iteration -1 and Φ at all other times. If Ψ is computationally expensive it is then only used at iteration -1. Figure 2.1 shows how this works.

Theorem 3 *For an irreducible, aperiodic chain with stationary distribution $f_X(x)$ (where the expectation of f_X exists) the above algorithm returns a sample with probability* 1 *from the stationary distribution of the chain.*

Proof

We first show that for a given realization of γ, with probability 1 the distance, under

the partial ordering, at time -1 between the minimum and maximum chains can be made arbitrarily small by starting the chains sufficiently far in the past. It is then shown that when starting far enough in the past these two chains will accept the same state at time 0. The state returned at time 0 will therefore be a sample from $f_X(\cdot)$ because all other possible chains are sandwiched between them.

For $t < -1$ define,

$$g_t(x) = \Phi(x, \gamma_t)$$

and

$$g_{-1}^t(x) = (g_{-2} \circ g_{-3} \circ \cdots \circ g_t)(x)$$

which is the function that returns the state of a chain at time -1 started from state x at time t. We therefore need to show that with probability 1

$$d_t = g_{-1}^t(x_{\max}) - g_{-1}^t(x_{\min}) \to 0 \text{ as } t \to -\infty.$$

Defining μ to be the expectation of a random variable with density $f_X(\cdot)$ we have

$$\begin{aligned} \mathbb{E}(d_t) &= \mathbb{E}(g_{-1}^t(x_{\max})) - \mathbb{E}(g_{-1}^t(x_{\min})) \\ &\to \mu - \mu \\ &= 0 \quad \text{as } t \to -\infty \end{aligned}$$

because $g_{-1}^t(x)$ converges in distribution to $f_X(\cdot)$ independent of x as $t \to -\infty$ and both chains are irreducible and aperiodic. Now, the d_t are non-negative and decreasing so their limit

$$d_{-\infty} = \lim_{t \to -\infty} d_t$$

exists and $d_{-\infty} \geq 0$. By the Monotone Convergence Theorem we have that $\mathbb{E}(d_{-\infty})$ exists and

$$0 \leq \mathbb{E}(d_{-\infty}) \leq \lim \mathbb{E}(d_t) = 0,$$

so $\mathbb{E}(d_{-\infty}) = 0$ and therefore $d_{-\infty} = 0$ a.s.. The distance between the minimum and maximum chains at time -1 thus converges to 0 almost surely as the chains are started further in the past.

Now consider two subsets of the partition and the point s defining the edge of them. It could be that the minimum and maximum chains converge to the point s, so that the two chains always end up in different sets of the partition at time 0. However with probability 0, $s = \lim g_{-1}^t(x_{\min})$ as $t \to -\infty$. So if we go back far enough both chains will accept the same state at time 0 with probability 1. □

As well as providing a way to decrease the use of difficult to perform updating steps this perfect sampling scheme allows extra flexibility in the choice of updating methods. The transition function Φ is used to bring chains closer together so that Ψ need only be concerned with mapping subsets of the state space to the same state. For example the partitioned versions of the algorithms from Green & Murdoch (1999) might be able to be used, but here we do not need to partition the state space into a finite number of subsets as required by Green & Murdoch's (1999) method. The partition only needs to have a minimum sized subset so that two chains getting closer together will eventually reach the same subset. The problems caused by the tails of distributions when using Green & Murdoch's (1999) method on its own could then be alleviated.

2.5.3 A Particular Case

We now go on to define Φ for the special case where $\mathcal{X} = (a, b)$ for $a, b \in \mathbb{R} \cup \{\pm\infty\}$, $l(\cdot)$ is a monotone function, and we can sample directly from $\pi(\cdot)$ using its inverse cumulative distribution function (CDF), $\Pi^{-1}(\cdot)$. We show that it is easy to construct a monotone transition function which uses two $U(0,1)$ random variables ϵ and w, so here we have $\gamma = (\epsilon, w)$. We use the method given in Devroye (1986) for sampling from $\pi(\cdot)$ truncated to an interval (a, b). Given its CDF $\Pi(\cdot)$ and a $U(0,1)$ variable w, the value

$$\Pi^{-1}(\Pi(a) + w(\Pi(b) - \Pi(a))) = \Pi^{-1}(w\Pi(b) + (1 - w)\Pi(a)) \tag{2.7}$$

is a sample from $\pi(\cdot)$ truncated to (a, b). The transition function $\Phi(x, \gamma)$ is constructed by first sampling a value for the auxiliary variable u by taking

$$u = \epsilon l(x)$$

and then using w and (2.7) to sample from $\pi(\cdot)$ restricted to the set $\{x : u < l(x)\}$ which will be an interval. We now show that this defines a monotone transition function by considering the two cases where $l(\cdot)$ is increasing and $l(\cdot)$ is decreasing.

$l(\cdot)$ Increasing

When $l(\cdot)$ is increasing then

$$\{x : u < l(x)\} = (l^{-1}(u), b)$$

and the next state of a chain currently at x can be simulated by returning the value

$$f_I(x) = \Pi^{-1}(w\Pi(b) + (1-w)\Pi(l^{-1}(u))) \tag{2.8}$$
$$= \Pi^{-1}(w\Pi(b) + (1-w)\Pi(l^{-1}(\epsilon l(x)))) \tag{2.9}$$

which is an increasing function of x because every element, $l(\cdot)$, $l^{-1}(\cdot)$, $\Pi(\cdot)$, and $\Pi^{-1}(\cdot)$, is an increasing function. So using (2.9) will produce a monotone chain because the partial ordering is preserved as

$$x \preceq y \iff l(x) \leq l(y) \iff x \leq y$$

and so for $x \leq y$ we have that $f_I(x) \leq f_I(y)$ as $f_I(\cdot)$ is increasing.

$l(\cdot)$ Decreasing

Here we have that

$$\{x : u < l(x)\} = (a, l^{-1}(u))$$

and so given a chain at x we return

$$f_D(x) = \Pi^{-1}(w\Pi(l^{-1}(\epsilon l(x))) + (1-w)\Pi(a)) \tag{2.10}$$

as its next state. The composition $l^{-1}(\epsilon l(x))$ forms an increasing function and so $f_D(x)$ is also an increasing function and thus preserves the partial ordering as

$$x \preceq y \iff l(x) \leq l(y) \iff x \geq y$$

and for $x \geq y$ we have $f_D(x) \geq f_D(y)$.

2.5.4 Truncated Gamma Distributions

The need to sample from truncated distributions can arise in a number of circumstances including constrained parameter problems, Gelfand, Smith & Lee (1992) and as conditional distributions of Gibbs samplers after the introduction of auxiliary variables, Damien, Wakefield & Walker (1999). So in this section we look at how well our perfect slice sampler performs at sampling from truncated Gamma densities. Let $f_X(x) \propto x^{c-1}\exp(-dx)$ for $x, c, d > 0$, $c \neq 1$ and assume that we wish to sample from $f_X(\cdot)$ restricted to the interval (a, b) where $0 \leq a < b < \infty$. We look at the perfect slice samplers induced by the following choices of $f_{XU}(x, u)$,

$$f_{XU}(x, u) \propto x^{c-1} I[u < \exp(-dx)] I[x \in (a, b)] \tag{2.11}$$

which can be obtained by using $\pi(x) = x^{c-1}$ and $l(x) = \exp(-dx)$, and

$$f_{XU}(x,u) \propto \exp(-dx) I[u < x^{c-1}] I[x \in (a,b)]. \tag{2.12}$$

which can be generated by choosing $\pi(x) = \exp(-dx)$ and $l(x) = x^{(c-1)}$. There is a slight restriction because for $c < 1$, x^{c-1} is not bounded above in the interval $(0,b)$, so (2.12) cannot be used in such cases. Both of these samplers satisfy the requirements of the perfect simulation algorithm from Section 2.5.3 and so it can be used here.

We now compare the performance of the perfect slice samplers for sampling from truncated Gamma distributions to another recent algorithm for sampling from truncated Gamma densities by Damien & Walker (2001), the AURS (adaptive uniform rejection sampling) scheme. Damien & Walker (2001) also split $f_{XU}(x,u)$ into the factorisation

$$f_{XU}(x,u) = \pi(x) I[u < l(x)]$$

and require that $\pi(\cdot)$ is a density from which it is relatively easy to sample from truncated versions and that $l(\cdot)$ is a bounded function. They then consider the method of first sampling a value u from the marginal distribution $f_U(u)$ and then a value x from the conditional distribution $f_{X|U}(x|u)$. The marginal distribution for U is a decreasing function over a bounded interval and so this can be sampled using adaptive rejection sampling but with Uniform segments in the envelope.

The time taken by three different schemes to obtain 10 000 000 samples from a number of truncated Gamma densities is shown in Table 2.1. The table gives the results from only a selection of example densities and the timings are subject to the efficiency of the programs running each algorithm. Overall the perfect slice samplers perform as well as the AURS algorithm, and both methods are quicker for individual examples. In particular the perfect slice samplers are better at sampling from distributions truncated to regions away from the modes of the distribution. This example serves to show that a CFTP algorithm can be constructed that is competitive with a conceptually simple rejection method, even though the complex machinery of CFTP would suggest that this might not be the case.

2.6 A Simple Bayesian Example

Damien et al. (1999) give examples of using auxiliary variables to simulate from non-conjugate Bayesian models using the Gibbs sampler. Here we look at obtaining perfect samples from a slightly more complicated version of one of their simple examples using the perfect polar slice sampler.

c	d	a	b	mode	AURS	PSS1	PSS2
0.1	0.1	1.5	2	1	43	99	103
0.1	0.1	6.5	7	1	43	99	101
0.1	2.1	1	2	0.05	171	183	117
0.1	2.1	6	7	0.05	584	200	103
1.1	2.1	0	1	0.52	98	203	95
1.1	2.1	6	7	0.52	594	206	94
10	5	2	4	2	1158	2388	1747
10	5	6	7	2	2036	512	225
20	2	10	11	10	845	273	214
21	20	1	2	1.05	2432	8309	7021
30	31	7	8	0.97	4099	1410	957

Table 2.1: Time in seconds taken to obtain samples from a number of truncated Gamma densities. The column headed PSS1 refers to the slice sampler defined in (2.11) and PSS2 refers to the one in (2.12).

We consider a Bernoulli regression for which we have data w_i such that $w_i \sim$ Bernoulli(p_i) where $p_i^{-1} = 1 + \exp(-\mu - xz_i)$ and z_i is a known explanatory variable. We place $N(0, \sigma^2)$ independent priors on the variables x and μ for some fixed value of σ^2. The posterior density for x and μ is then

$$f_{XM}(x,\mu) \propto \exp\left(-\frac{x^2}{\sigma^2}\right) \exp\left(-\frac{\mu^2}{\sigma^2}\right) \prod_{i=1}^{n} l_i(x,\mu)$$

where $l_i(x,\mu) = (1+\exp(-\mu - xz_i))^{-w_i}(1+\exp(\mu + xz_i))^{w_i - 1}$. Following the method in Section 2.4 we introduce an auxiliary variable U with the joint distribution

$$f_{XMU}(x,\mu,u) \propto \exp\left(-\frac{x^2}{\sigma^2}\right) \exp\left(-\frac{\mu^2}{\sigma^2}\right) I[u < l(x,\mu)]$$

where $l(x,\mu) = \prod_{i=1}^{n} l_i(x,\mu)$. It is clear that $l(x,\mu)$ is bounded above, because for each i, $l_i(x,\mu)$ is a product of two values each less than 1, and that $\pi(x,\mu) = \exp(-\frac{x^2}{\sigma^2})\exp(-\frac{\mu^2}{\sigma^2})$ is proportional to an independent bivariate Normal, and it is possible to sample from truncated versions of this distribution. The $w_{t,j}$'s needed in the perfect slice sampling are generated using the conditional distribution $f_{XM|U}(x,\mu|u)$ which is proportional to

$$\exp\left(-\frac{x^2}{\sigma^2}\right) \exp\left(-\frac{\mu^2}{\sigma^2}\right) I[u < l(x,\mu)].$$

In order to look at the performance of this perfect sampler we generated $n = 10$ values from the likelihood using a fixed choice of x and μ to serve as a test dataset. We then applied the perfect slice sampling algorithm to this dataset. To generate the samples for the sequences of $w_{t,j}$'s required by the algorithm we needed to generate samples from the truncated distributions $f_{XM|U}(x, \mu|u)$. To do this we used rejection from the prior because the level sets are difficult to find. The efficiency of using the prior to generate samples from the conditional distributions depends of the choice of σ^2 specified in the prior. If σ^2 is large then the prior is very over-dispersed and the expected number of samples required until the condition $u < l(x, \mu)$ is satisfied will be high. We found that for values of σ^2 up to about 200 it is practical to use our perfect slice sampling algorithm, with rejection from the prior, to generate samples from the posterior with the algorithm taking a few seconds. For σ^2 greater than 200 however the algorithm becomes too slow. If we were to use perfect slice sampling as a general solution to sampling from this model then it would be necessary to find convenient supersets of the level sets $\{x, \mu : u < l(x, \mu)\}$ to use as rejection envelopes when sampling from $f_{XM|U}(x, \mu|u)$.

2.7 Can Perfect Slice Sampling be Extended Further?

The splitting of a density into a product of two terms used in Section 2.4 suggests the possibility of extending perfect slice sampling to more general distributions arising from Bayesian models. In general the Bayesian posterior distribution of a random variable X consists of the product of the prior and likelihood. So in (2.4) we can take $f_X(x)$ to be the posterior, $\pi(x)$ the prior, and $l(x)$ the likelihood. Then if these satisfy the conditions required by the perfect slice sampler we have an algorithm that can get a perfect sample from the posterior. The first condition is that we can sample from truncated versions of the prior. If the prior is proper then we may well be able to do so, certainly if it is specified using a directed acyclic graph (DAG) then we can sample from it. However sampling from the prior truncated to the level sets $\{x : u < l(x)\}$ is likely to be difficult and slow because by its nature the prior should be over dispersed compared to the likelihood, and this is especially true in high dimensions. If some parameters appear in the prior but not the likelihood then this also causes problems as the level sets do not contain all the parameters which are sampled from the prior. The second condition is that the likelihood must be bounded above and that we can numerically find the maximum, which is satisfied if there is a maximum likelihood estimate, and this is often the case for models specified using DAGs. We note that there may be other factorizations which lead to π being a density and l being bounded

above and so choosing the prior and likelihood as the two terms might not be the most efficient way.

If perfect slice sampling is going to be used in practice then we probably need to incorporate the ideas from Section 2.5 and try to find methods that use the conditional distributions to update chains but that preserve the partial ordering, which may be possible if the conditional distributions have certain scaling properties. We have shown how for the coalescing step at time 0 we do not necessarily need to find a transition function that updates the state space to a finite number of values. This is the case for most of the current perfect simulation algorithms that cope with continuous state spaces. In our algorithm we only need a method that coalesces chains that get close enough to each other in the partial ordering.

Chapter 3

Approximate Coupling from the Past

3.1 Introduction

Despite being clearly appealing, (efficient) perfect simulation algorithms are not available to sample from the distributions arising from most statistical models which consist of mainly continuous parameters. Perfect simulation algorithms tend to require the distribution of interest to have certain properties or be reasonably tractable. There are many distributions which do not meet the requirements of current perfect simulation algorithms, some examples of difficulties in current methods are the following. A suitable over-dispersed distribution cannot be found for use as a Metropolis-Hastings proposal in perfect simulation algorithms such as those of Corcoran & Tweedie (1998) and Murdoch & Green (1998). Following the set of states that do not accept the proposed new state at each iteration is intractable as required by algorithms from Breyer & Roberts (2000) and Murdoch & Green (1998). There is no partial ordering, monotonicity, or upper and lower states as needed by Mira et al. (2001), and although the perfect polar slice sampler makes perfect simulation possible for a much wider class of distributions it is not necessarily efficient.

We propose to use the idea of coupling from the past in an algorithm that will return a sample from close to the desired distribution (under certain assumptions), which avoids the need for the distribution to have special properties or be sufficiently tractable that specialist algorithms can be used. The general idea is to run a finite number of chains updated using the same set of random variables, which we will refer to as coupled, from sufficiently far in the past that they all coalesce to one state at time 0. Let T denote this time in the past. The initial states of these chains should

be suitably chosen so that there is a high probability that they reach the same state at time 0 as a chain started infinitely far in the past, such a chain is stationary. The common state at time 0 would then be a sample from the desired distribution.

If the method used to update chains is taken from a perfect simulation algorithm, so chains started from every state are guaranteed to eventually coalesce, then it is possible to find the unique state which the hypothetical chain started infinitely far in the past reaches at time 0. However the methods used to update the chains will most likely not be taken from a perfect simulation algorithm, and so will be constructed in a way that means it is not possible to find the state which a chain started infinitely far in the past reaches. In this situation the performance of ACFTP can be assessed by considering the probability of the ACFTP chains reaching the same state at time 0 as a chain started in a state sampled from the stationary distribution at time T.

The procedure is described fully in Section 3.2. We also provide an approximate bound on the total variation distance between the distribution of the state at time 0 and the desired distribution, which is given in Section 3.3. This procedure, which we will call approximate coupling from the past (ACFTP), replaces the need for a burn-in period by automatically presenting the state at time 0 as a sample from a distribution sufficiently close to the desired one.

This idea is similar to that of Johnson (1996) whose idea is to run multiple coupled chains forward in time. In Johnson's (1996) method the chains should be started from an over-dispersed distribution as defined by (1.13). The iteration at which the chains coalesce is noted. This is then repeated with different random variables, for the same number of chains, to obtain an estimate of the distribution of the iteration at which the chains coalesce. Based on the quantiles of this distribution Johnson (1996) gives a bound on the total variation distance between the state of a single chain started in the over-dispersed distribution and its stationary distribution at a given iteration. However, the state of the chains when they coalesce will not be a sample from their stationary distribution. This point was noted by Propp & Wilson (1996) and Johnson (1996). The distribution of the state that the chains coalesce in depends on both the stationary distribution and the choice of update. To demonstrate this Propp & Wilson (1996) give the example of a chain where some states have a unique predecessor; such states will never be returned by an algorithm stopping when chains coalesce.

Both Johnson's (1996) diagnostic and the ACFTP procedure require methods that coalesce chains in a continuous space. Johnson (1996) considers two chains to have coalesced if the absolute distance between each univariate component of the two chains is less than a suitably chosen small value. The two chains are then taken as having reached the same state and only one chain is followed thereafter. For the problems

considered by Johnson (1996) the inverse of the cumulative full conditional distributions are available from which simulation is easy using one $U(0,1)$ random variable. With each chain using the same random variable at each iteration Johnson (1996) observed that chains tended to get closer together over time. This way of coalescing chains does not always extend so well to other situations, for example the natural way to couple Metropolis-Hastings chains is for the chains to use the same random variables to generate the proposals and then the same random variable in the acceptance step. But as Johnson (1996) points out, even when the acceptance probability for two chains is close, a uniform random variable falling between the two probabilities can separate the sample paths by an arbitrarily large distance.

If chains are coupled so that they can coalesce exactly, when two chains coalesce they remain together for all future iterations. Methods that coalesce chains exactly by using independent proposals on different subsets of the state space are looked at in Section 4.2.

3.2 The ACFTP Procedure

Consider the situation in which we wish to sample from the distribution π on the state space $\mathcal{S}$ and have chosen a Markov chain that is believed to be good at simulating from π. In order to describe the ACFTP algorithm we first define some notation. We use the stochastic recursive sequence (SRS) representation of a Markov chain from Section 2.3. Let $\gamma = \{\gamma_t\}_{t=-\infty}^{\infty}$ be a sequence of sequences of independent random variates, so each γ_t is itself a sequence of independent random variates. Let the current state of the Markov chain be x_t, then the next state of the chain is given by,

$$x_{t+1} = \Psi(x_t, \gamma_t)$$

where $\Psi(\cdot,\cdot)$ updates a Markov chain according to the chosen transition kernel. There are many ways to choose the function $\Psi(\cdot,\cdot)$ that will preserve the transition kernel of the Markov chain. Suitable choices for Ψ that allow chains from a continuous state space to coalesce exactly are discussed in Section 4.2. Once Ψ is chosen the ACFTP procedure proceeds as described below.

For convenience we define the function $f(\mathcal{A}, \gamma, T)$ where $\mathcal{A}$ is a set of states in $\mathcal{S}$, γ is a sequence of sequences of $U(0,1)$ variates (as defined earlier) and $T \in \mathbb{Z}$ with $T < 0$. Given these inputs we can consider the result of running stochastic recursive sequences starting from each state in $\mathcal{A}$ at time T using the random variates in γ, updating them using the Ψ function until time 0. Let f be the function that returns the state at time 0 if all the SRSs coalesced and FAIL otherwise.

The ACFTP algorithm consists of four steps. The general idea behind each step is now described.

Step 1. Choose a set of states $\mathcal{A}$ such that if chains starting from these states are run from sufficiently far in the past that they all coalesce, it is believed there is a high probability that they reach the same state at time 0 as a chain started in a state sampled from π at the same time in the past. The choice of suitable starting states to include in the set $\mathcal{A}$ is discussed in Section 3.4.

Step 2. Find a time in the past T such that chains starting from the set of states in $\mathcal{A}$ reach the same state x_0 at time 0.

Step 3. Generate some samples by running a chain forward in time starting in the state x_0. These samples should come from a distribution close to π if the assumptions in the first step hold.

Step 4. Start chains from these sampled states at time T. These chains are used to check whether the assumption made in the first stage, about a high probability that a stationary chain reaches x_0 at time 0, holds.

When implementing ACFTP, if all of the chains in Step 4 reached x_0 at time 0 then x_0 is returned as a sample from π. If not then the set of states in $\mathcal{A}$ can be expanded to include the samples from Step 3, and the algorithm restarted from Step 2. The initial choice of $\mathcal{A}$ and the use of the checking chains in Step 4 means that we should be confident that the value x_0 eventually returned is a sample from π.

It may be that the particular sequence of γ which is used leads to chains starting in the set of states $\mathcal{A}$ coalescing unusually quickly. To provide some protection against this when running ACFTP in practice we recommend using a few copies of the procedure described above, each using an independent set of γ and running the chains in each copy from sufficiently far in the past that the chains within every copy coalesce.

Further security against chains not mixing throughout the state space can be obtained by using the forward samples generated in Step 3 in every copy of the procedure. It may happen that chains in one particular copy do not visit an important part of the state space. To be confident that there is a high probability that a stationary chain coalesces with the ACFTP chains in this copy it would be a good idea to include a chain starting from the unvisited part of the state space, this is done by including in the starting states in Step 4 the forward samples from every copy.

Before we describe the algorithm in full as used throughout the rest of this thesis we first define some notation which is used in the description of ACFTP. We assume that r repetitions of the procedure described above are used, each using an independent set of random variables $\gamma^{(j)}$ for $j = 1, \ldots, r$. In Step 1 each set of starting states $\mathcal{A}_j$ contains m samples drawn from a starting distribution denoted by p_0. In Step 3 each

repetition collects s samples from its forward chain. The ACFTP algorithm which is used in the rest of this thesis is given by the following algorithm.

Step 1.

For each j from 1 to r

Generate m samples from p_0 and let $\mathcal{A}_j$ consists of these m samples.

Set $T = -1$ and $x_0^{(j)} =$ FAIL for all j.

Step 2.

While there exists an $x_0^{(j)} =$ FAIL repeat

Set $T = 2 * T$.

For each j from 1 to r,

Let $x_0^{(j)} = f(\mathcal{A}_j, \gamma^{(j)}, T)$.

Step 3.

For each $j = 1 \ldots, r$,

Run a chain forward from state $x_0^{(j)}$ and collect s samples spaced h iterations apart.

Add these s samples to all $\mathcal{A}_k$ for $k = 1, \ldots, r$.

Step 4.

For each j from 1 to r,

Let $x_0^{(j)} = f(\mathcal{A}_j, \gamma^{(j)}, T)$.

If there does not exist an $x_0^{(j)} =$ FAIL,

Then return $x_0^{(1)}, \ldots, x_0^{(r)}$ as independent samples from π.

Otherwise,

Return to Step 2.

If the distribution p_0 is well chosen then after the first time Step 2 is performed the states $x_0^{(1)}, \ldots, x_0^{(r)}$ should be samples from a distribution close to π. Instead of checking that chains from all possible states have coalesced as in the perfect sampling algorithm of Propp & Wilson (1996), we check whether chains starting from most states would have coalesced, where most is in the sense of their total probability distribution under π. If most chains coalesce then we can regard the one state they coalesce to as a sample from a distribution close to π.

Step 3 is used to generate what we hope are approximate samples from π, because under the assumption that $x_0^{(1)}, \ldots, x_0^{(r)}$ are samples from π each of the forward chains will be simulating from π. These samples will be used in a check on the procedure in the next step. More details on exactly how to get these samples are given in Section 3.5.

The idea behind Step 4 is to check how likely a chain starting from a sample from π at time T would be to coalesce with the ACFTP chains, and hence whether we can

regard $x_0^{(1)}, \ldots, x_0^{(r)}$ as samples from π. The samples from Step 2 are used in place of actual samples from π, and if all the samples from Step 2 coalesce then $x_0^{(1)}, \ldots, x_0^{(r)}$ can be regarded as samples from π.

With a good choice of p_0 the assumptions and ideas behind ACFTP should hold and the checking chains should coalesce to $x_0^{(1)}, \ldots, x_0^{(r)}$. If the checking chains do not coalesce with the other chains, when started at time T, this shows that the forward chains reached previously unvisited parts of the state space, and that a significant proportion of chains starting from states sampled from π would not coalesce by time 0. The previously unvisited parts of the state space are included in ACFTP by restarting Step 2, but including the samples from the forward chains in the set of starting states.

When actually implementing Step 4 we note that for each repetition j it is unnecessary to follow all the chains in the set $\mathcal{A}_j$. Only the most recently added states need to be followed along with one chain that reaches $x_0^{(j)}$. These chains may well coalesce before time 0 in which case the checking can stop early as all the chains will then go on to reach $x_0^{(j)}$.

3.3 An Error Bound

For each repetition j, ACFTP provides an approximate error bound on the total variation distance between the distribution of $x_0^{(j)}$ and π. We can obtain such a bound by considering a hypothetical chain sampling exactly from π, and the probability of it coalescing with the chains used in ACFTP. For notational convenience we remove the superscript referring to the particular repetition under consideration, as the bound obtained is the same for each repetition.

To construct the error bound we consider the situation in which γ is fixed and a state x_0 and a time, $T < 0$, are presented such that it is believed that there is a high probability of a chain starting from a state sampled from π at time T reaching x_0 at time 0, if it is updated using the same γ.

In order to test the belief that this probability is high c test chains could be started at time T in states sampled from π, and it can be observed whether they all reach x_0. In practice this test is performed using chains starting in states sampled from a distribution believed to be close to π. To calculate the error bound we consider the distribution of x_0 in terms of the probability that a chain started at time T in a state sampled exactly from π reaches x_0 given that we stop ACFTP when the c test chains have reached x_0.

This idea is now formalised and we describe how to construct the error bound in terms of the steps in the ACFTP procedure.

At the end of Step 2 of the ACFTP algorithm a time T and a state x_0 is produced. It is hoped that through a good choice of the starting states generated in Step 1, there is a high probability that a coupled chain starting at time T from a state sampled from π will reach x_0 at time 0. This hypothetical chain is generated using the same Ψ and γ as the chains used in ACFTP. Let $\tilde{z}_T$ denote the random state sampled from π which serves as the stationary chain's starting state at time T. The state of the stationary chain at time 0 is therefore given by $f(\{\tilde{z}_T\}, \gamma, T)$ and this is also a sample from π. To obtain our error bound we define

$$\varepsilon_T = \mathbb{P}(f(\{\tilde{z}_T\}, \gamma, T) \neq x_0 | \gamma, T, x_0)$$

at the end of each iteration of Step 2.

In Step 3 the starting states for some test chains are generated, which if the starting states of the chains in Step 1 were well chosen should be close to being samples from π. These samples should also be close to independent because they are taken from sufficiently far apart (which is discussed in Section 3.5) and pooled from every repetition. Under the assumption that the rs test samples are independent samples from π then when we run Step 4 of ACFTP we have that

$$\begin{aligned} \mathbb{P}(\text{All the test runs lead to } x_0 \text{ but } f(\{\tilde{z}_T\}, \gamma, T) \neq x_0 | \gamma) &\simeq (1 - \varepsilon_T)^{rs} \varepsilon_T \\ &\leq (1 - \frac{1}{rs+1})^{rs+1} \frac{1}{rs} \\ &< \frac{1}{rse} . \end{aligned} \tag{3.1}$$

The methods used to obtain these inequalities are given in Appendix A. For a chosen value of δ we can now make the total variation distance between the distribution of the value x_0 returned and π less than δ by taking the following strategy.

From the first pass through Step 2 obtain a time T. In Step 3 generate s_1 samples from the forward runs where

$$\frac{1}{rs_1 e} \leq \frac{\delta}{2}$$

and if all the test chains coalesce accept x_0 as a sample from π. If not then repeat Step 2 using the extra starting states obtained from the forward runs to decrease the value of T. In Step 3 now generate s_2 test samples from the forward chains such that

$$\frac{1}{rs_2 e} \leq \frac{\delta}{4} .$$

If all the chains coalesce then accept their common value at time 0 as a sample from

π. If not then continue this procedure.

The probability that a stationary chain started from the final choice of T does not coalesce with the ACFTP chains is therefore less than δ no matter how many attempts are made to find a suitable value of T.

We do not use this error bound directly when demonstrating ACFTP in the next chapter, instead we use the idea underlying the error bound to show that ACFTP should achieve a much smaller value of the total variation distance between the sample returned by ACFTP and π than the calculated bound. In the next chapter we estimate directly the probability of a stationary chain coalescing with the ACFTP chains by taking samples from π and following chains starting from these states.

3.4 Choosing Initial Starting States

Any diagnostic method that assesses the convergence of a Markov chain needs to be given a starting value for the chain, and possibly multiple starting values if many chains are used. Gelman & Rubin (1992) suggest a way to construct an over dispersed distribution as follows.

First find the modes using some optimisation algorithm and then estimate the second derivative matrix at each mode, which is used to construct a mixture of multivariate Normals approximating our target distribution. This choice is then converted to a mixture of multivariate t distributions with the same mean and covariance matrices with a suitably chosen small number of degrees of freedom, (Gelman & Rubin (1992) suggest 4). The starting states for each chain are drawn from this mixture of t distributions using importance resampling, where initially a large set of values is sampled from the mixture of t distributions and then the desired number of states sampled from this set with the probability of each state being drawn proportional to the ratio of the desired density divided by the density of the mixture of t distributions.

It is of course sensible after all of this effort to make sure that if multiple chains are used that one chain is started in each mode. When one (significant) mode is present Gelman & Rubin (1992) suggest using 10 chains and increasing this when more than one mode is present, but it seems to be more common in practice to use around five. Gelman & Rubin (1992) themselves acknowledge that this elaborate and time consuming method of choosing starting states is not necessary for simple problems. However if it is suspected that there may be more than one significant mode or that the chain may not mix well throughout the state space then it is important to use a good over-dispersed distribution for choosing the initial states of chains.

In practice when choosing the starting states for ACFTP we would recommend using

this method, or at least any reasonable variation of it. The examples looked at in this thesis are however well studied and so we do not go through Gelman & Rubin's (1992) procedure. Instead for the multivariate Normal distributions we look at we simply take the starting states from another more dispersed multivariate Normal distribution. For the distributions arising from Bayesian models, including the mixture distributions, we use the prior as our starting distribution. By its nature the prior should not be a bad starting point for constructing an over dispersed distribution because if a significant proportion of the posterior distribution was in an area of relatively low prior probability then we would be questioning our choice of prior. We will also see that ACFTP performs well for even very badly chosen starting distributions consisting of a distribution with point mass.

3.5 Collecting Samples from the Forward Chains

To obtain the s samples from the forward runs we need to choose how far apart these samples should be taken. The idea behind the error bound is that these samples should be roughly independent samples from the stationary distribution of the chain. Running the chains from the past will have provided a time T such that in each repetition all of the chains in the test sets have coalesced. The sample at time 0 in each repetition should therefore be effectively independent of the starting state of each chain. This suggests that it takes around $-T$ iterations for a chain to forget its current state. So for the forward runs we suggest taking the s samples $h = -T$ iterations apart and including the state at time 0. This can also be used when producing samples for the final run as an indication of how many iterations it takes to get independent samples.

We also mention that a new seed for the Uniform random number generator was used each time the forward chains were run. It may be that when Step 4 of ACFTP fails and Step 2 is implemented again the samples $x_0^{(1)}, \ldots, x_0^{(r)}$ are the same as those obtained from the previous iteration of Step 2. If the same seed were used to generate the forward samples, the forward samples obtained would be the same as before and so not provide any checking capability.

3.6 A Final Check

It is also possible to do a final check after ACFTP has finished and chains have been run forward to produce samples from π which are to be used in estimating quantities of interest. States can be taken from the r forward runs and chains starting from them run from time T to time 0 using the same random variables at each iteration as

used by ACFTP. This can be done for each repetition and if an unsatisfactorily low proportion of these chains coalesce to $x_0^{(1)}, \ldots, x_0^{(r)}$, (for the appropriate repetition) then the simulation of states from π should be looked into.

Chapter 4

Applying Approximate Coupling from the Past

4.1 Introduction

All perfect simulation algorithms require methods that enable two chains to coalesce to exactly the same state, so that one state can be returned at time 0. ACFTP has the same requirement. For chains with a continuous state space this creates a substantial problem as chains cannot just be left to coalesce on their own as they can in discrete cases. Some existing methods for doing this in perfect simulation algorithms were described in Section 1.7. However perfect simulation algorithms need coupling methods that guarantee the perfect nature of the algorithm, for example the transition function might have to be monotone, or the set of states not accepting a proposal followed at each iteration. ACFTP does not need such special properties and so the methods for coalescing chains for use with ACFTP can be chosen with greater flexibility. One major advantage is that standard transition kernels such as the Gibbs sampler can be used in conjunction with the transition functions that coalesce chains to improve the mixing of the chains and thus speed up coalescence. This means that the transition functions that coalesce chains do not also have to mix well. In Section 4.2 we look at some straightforward methods for coalescing chains in a continuous space which could be used with ACFTP.

We note that, as used by Guglielmi, Holmes & Walker (2001) any algorithm implemented in practice will be done so on a computer, which is a finite state machine, so two chains should eventually coalesce without any need for special methods. The methods described in this section improve upon relying on machine precision to coalesce chains.

In the remainder of this chapter we apply the ACFTP procedure to some example

distributions in order to show how well it performs and support the recommended choices for the parameters which need to be chosen. For these parameters we will use the same notation as before, the number of chains to be started from the starting distribution in each repetition m, the number of replications r, and the number of samples obtained by the forward runs s. ACFTP is also compared to some of the popular currently used diagnostics in Section 4.5.

Assessing the performance of ACFTP is easily done directly. The underlying idea behind ACFTP is that once it has finished and produced a sample at time 0, call it x_0, then if a chain was started from the stationary distribution at the time in the past that ACFTP went back to and is updated using the same random variables, this stationary chain will be very likely to reach x_0 at time 0. In this section we look at the performance of ACFTP on a number of distributions. If the proportion of times that the stationary chain coalesces with the original chains in ACFTP is near 1, then the ACFTP procedure is producing samples from or very near to the stationary distribution. We will see that the error bound from Section 3.3 holds and that ACFTP does a lot better. This is a direct way to check the assumptions in the ACFTP theorem using the machinery of ACFTP, which you cannot do with other diagnostic methods. In the examples looked at here it is only the multivariate Normal for which we can generate samples from the stationary distribution. In the other examples we substitute samples from very long runs as samples from the stationary distribution. The extremely long length of these runs will allow us to assume that these are indeed samples from the stationary distribution.

To demonstrate the performance of ACFTP we apply it to some multivariate Normal distributions in Section 4.3 and then go on to look at a couple of distributions arising from Bayesian models in Section 4.4. Unlike other diagnostics, including the methods described in Cowles & Carlin (1996), ACFTP can be used to assess the convergence of chains simulating from variable dimension problems and in Section 4.7 we apply ACFTP to some mixture distributions.

4.2 Methods for Coalescing Chains in a Continuous State Space

4.2.1 Johnson's Coupling Method

Johnson (1996) couples chains so that at each iteration, when updating a particular component every chain uses the same random variable. He generates the next state of a chain using one of two methods. The first is to use the inverse cumulative distribution

function of each component, in which case the random variable used at each iteration is a $U(0,1)$. The second method is to use scaling and shifting of the random variable, in which case the random variable would be a sample from a standard distribution family member from which the conditional distributions belong, for example a $N(0,1)$ distribution if the conditional distributions are Normal. We will see later that this method can be generalised to cope with a wider range of distributions. Johnson (1996) observed that in his examples chains coupled in this manner tended to get closer and closer together and so he proposed the following method for coalescing chains.

He defined two chains with states in $\mathbb{R}^n$ to have coalesced if every component of each chain is within ϵ of each other, for some suitably chosen small value of ϵ. To avoid Johnson's (1996) diagnostic prematurely diagnosing convergence, ϵ should be chosen in the belief that if two chains are within ϵ of each other then they will remain so thereafter. Johnson (1996) looked at mixtures of bivariate Normals with variances in the range 0.06 to 0.25 and at the hierarchical model for rat growth which will be looked at in Section 4.4.2, (although with a slightly different specification of the model and updating some of the parameters jointly). For these examples he uses a value of $\epsilon = 0.001$ for all parameters but does not give any guidance on choosing ϵ when approaching a new problem. In general choosing a suitable value for ϵ is not so straightforward. If ϵ is chosen too small then chains will take a long time to coalesce and other methods of coalescing chains will be a lot faster. If ϵ is too large then chains could get within ϵ of each other but get further apart again later and if this occurs then using Johnson's (1996) coalescence criteria would prematurely diagnose convergence. The problem of choosing a suitable value can be avoided by using methods that coalesce chains exactly such as that recommended by Neal (2002).

4.2.2 Neal's Coupling Method

For the remainder of this Section we will use the same notation as that used in Section 1.6, so let x be the current state of a chain with desired stationary distribution π. If for different values of x but the same γ, Ψ is constructed so that it returns the same value, at least some of the time, then two chains getting closer together should coalesce to exactly the same state eventually. One way to produce such updates is by using a Metropolis-Hastings proposal which is partially independent of the current state of the Markov chain.

For a Markov chain simulating from $\mathbb{R}^d$, Neal (2002) achieves this by updating each univariate component of the chain in turn, using a Uniform proposal. The Uniform proposal is coupled in a such a way that two chains in different states will make the same proposal if they are close enough. Now let x denote the current state of component j

of a chain. The Uniform proposal works by partitioning the state space into intervals of width 2ω and a proposal is made to the centre of the interval that x lies in. This Metropolis-Hastings step requires two $U(0,1)$ random variables at each iteration so we let $\gamma = (u_1, u_2)$. The update step Ψ is,

$$\Psi(x,\gamma) = \begin{cases} f(x,u_1) & \text{if } u_2 < \frac{\pi(f(x,u_1))}{\pi(x)} \\ x & \text{otherwise} \end{cases}$$

where

$$f(x,u_1) = 2\omega\left\{\left(u_1 - \frac{1}{2}\right) + \left\lfloor \frac{x}{2\omega} - u_1 + 1 \right\rfloor\right\}. \tag{4.1}$$

Given the $U(0,1)$ variate u_1, f generates the random grid of points,

$$\ldots, -\omega + 2\omega u_1,\ \omega + 2\omega u_1,\ 3\omega + 2\omega u_1, \ldots$$

spaced 2ω apart, and returns the point nearest to x. The expression for f is derived in Appendix B. Two chains with the same nearest point in the grid will coalesce (in the current variable) if they both accept the proposal. For variables restricted to a subset of $\mathbb{R}$ the proposal is rejected if it lies outside the subset, although we could be more efficient and truncate the grid this should not be a significant problem.

This method of coalescing chains has also been presented by Wilson (2000*b*), who calls this a multishift coupler. Wilson (2000*b*) generalizes the idea to other distributions, such as the Normal distribution. Wilson (2000*b*) also shows how a multiscale coupler may be produced which can be used with distributions that have scaling properties, such as the Gamma distribution.

On its own this Metropolis-Hastings sampler would perform poorly for most problems and it will be desirable to use other methods for updating the Markov chain such as the Gibbs sampler, slice sampler, or other Metropolis-Hastings proposals among many other possibilities. So as suggested by Neal (2002) who uses Langevin updates, the Uniform coupling proposals should be used every so often to coalesce chains exactly and more efficient updating methods should be used to bring chains close together. In this thesis we consider schemes that use Neal's (2002) proposal on iterations t where $t \bmod L = l$ for a fixed choice of L and possibly more than 1 value of l, which attempts to coalesce chains every L iterations and possibly has a few attempts in a row. This seems a reasonable way to choose when to attempt to coalesce chains, although other schemes have not been looked at. We will also look at varying the choice of ω_j.

4.2.3 Coupling Through the Parameters of Conditional Distributions

We have also looked at another method for getting chains to coalesce which uses the conditional distributions directly to do the coalescing. It uses the same idea as Neal (2002), of partitioning the state space at each iteration and making the same proposal for chains lying in the same subset, but instead of using a Uniform proposal, the proposal is based on the conditional distribution.

Let the current state of a chain at time t, be $x = (x_1, \ldots, x_k)$ and suppose that the full conditional distribution of component j depends on a vector $\theta^{(j)} = (\theta_1^{(j)}, \ldots, \theta_\nu^{(j)})$ which is a function of $x_{-j} = (x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_k)$, for example $\theta^{(j)}$ might consist of the location and scale parameters of a Gamma distribution. Let $p(x_j|\theta^{(j)})$ denote the full conditional distribution of component j. We update x_j using a Metropolis-Hastings step with proposal $p(x_j|\check{\theta}^{(j)})$ where $\check{\theta}^{(j)} = (\check{\theta}_1^{(j)}, \ldots, \check{\theta}_\nu^{(j)})$ and

$$\check{\theta}_l^{(j)} = r_{j,l}(\theta_l^{(j)}) \qquad l = 1, \ldots, \nu.$$

Each function $r_{j,l}$ needs to be chosen so that it maps the space of $\theta^{(j)}$ onto a suitable countable set of values. Here we only consider cases where $\theta_l^{(j)} \in \mathbb{R}$ and choices for $r_{j,l}$ of the form

$$r_{j,l}(\theta_l^{(j)}) = \omega_{j,l} \left\lfloor \frac{\theta_l^{(j)}}{\omega_{j,l}} \right\rfloor + \frac{\omega_{j,l}}{2}.$$

This choice of $r_{j,l}$ partitions the state space of $\theta_l^{(j)}$ into intervals of width $\omega_{j,l}$ and then returns the centre value from the interval that $\theta_l^{(j)}$ lies in. Thus two chains whose j^{th} component has parameters of its conditional distribution in the same subset of $\mathbb{R}^\nu$, as partitioned by the $r_{j,l}$, will propose the same next state and if accepted the j^{th} component of these chains will coalesce.

It would also be possible to make the functions $r_{j,l}$ partition the state space randomly in the same way that Neal's (2002) method positions the intervals at random. The benefits of this were not investigated because, as will be discussed in Section 4.2.5, this method is less desirable than using Neal's (2002) method.

4.2.4 Other Methods

There are other schemes for coalescing chains in a continuous space which are not further considered here, but which we briefly mention. Some coupling methods depend on the current state of each chain (Markov maximal coupling and mixture coupling Reutter & Johnson (1995)). These are not suitable for use with ACFTP because the state returned by ACFTP would be entirely dependent on the starting states of the

Coupling through parameters			Neal's coupling	
ω_μ	ω_{σ^2}	mean	ω	mean
10^{-3}	10^{-6}	419	0.0015	401
10^{-2}	10^{-5}	392	0.015	321
10^{-1}	10^{-4}	448	0.15	257
1	0.001	875	1.5	2006
10	0.01	2632	15.0	2569

Table 4.1: Comparing coalescence times (from 1000 samples) for 2 chains sampling from a 6 dimensional multivariate Normal distribution, for various values of the parameters in the coupling methods.

chains used in ACFTP, and so when the conceptual chain sampling from π is included in the set of coupled chains the state returned at time 0 will change. Other coupling methods that are designed for use in perfect simulation algorithms (Murdoch & Green (1998), Breyer & Roberts (2000)) could be used with ACFTP but these tend to be less straightforward to use and require problem specific programming. Green & Murdoch (1999) introduced the bisection coupler that allows Metropolis-Hastings proposals to be constructed that generate a finite list of proposal values from a continuous bounded set of potential states. This method could be used with ACFTP to coalesce chains in a continuous space, but has not been used in this thesis.

4.2.5 Comparing Coupling Methods

If we were to follow Johnson's (1996) method and assume that once all the components of two chains have got within ϵ of each other, that they can be regarded as having coalesced then we have to choose suitable values of ϵ for each parameter. Suitable values for the ϵ should be based on the marginal stationary distribution of each variable and should be small compared to how spread out this distribution is. If the value of ϵ is chosen to be too big then convergence will be prematurely diagnosed. If ϵ is too small then we are getting toward using machine precision to coalesce chains and coalescence could take a long time. Choosing ϵ based on the stationary distribution is clearly not ideal as this is what we are trying to estimate and do not yet know. For this reason methods that coalesce chains exactly are preferred, and we will look at choosing the parameters used in the coupling method without having to look at coalescence times or have any knowledge of the stationary distribution. Coalescing chains exactly also avoids the approximation in Johnson's (1996) method and stops any concerns about chains getting further apart again once they are within ϵ of each other.

The methods of coupling through the parameters of the conditional distributions

and Neal's (2002) coupling method have been applied to the examples looked at in the rest of this chapter. As an example Table 4.1 gives results on the mean number of iterations taken for 2 chains to coalesce using different parameters in the coupling methods, when simulating from the 6 dimensional multivariate Normal described in Section 4.3.1 and referred to as MVN1. The mean has been given as a convenient summary of the distribution of coalescence times because the distribution of coalescence times followed the same pattern across different values of ω. Each distribution was right-tailed and the variance of the distribution increased roughly proportional to the increase in the mean.

For both coupling methods we used the same parameters in the coalescence step for each component. In the notation of Section 4.2.2, for Neal's (2002) coupling we take $\omega_j = \omega$ for $j = 1, \ldots, 6$ for some ω. For the coupling through parameters in Section 4.2.3 we have that the conditional distributions $p(x_j|\theta^{(j)})$ are Normal and so the parameters consist of interval widths for the mean and variance. We take ω_μ and ω_{σ^2} to be the width of the partitions for the mean and variance respectively for all 6 components. Generally and in this example in particular coupling through the parameters of the conditional distribution takes a little longer to coalesce chains than Neal's (2002) coupling method. Unfortunately both methods perform poorly if too large values of the coupling parameters are chosen, although coupling through parameters is slightly more robust to an overly large choice.

In Section 4.3.2 we give a convenient way to avoid the problem of a poorly chosen ω and so because Neal's (2002) coupling method is simpler to implement and computationally quicker we recommend the use of it with ACFTP, and this is the method used to coalesce chains throughout the rest of this thesis.

4.3 Applying ACFTP to the Multivariate Normal

4.3.1 Introduction

The multivariate Normal distribution can serve as a rough approximation to many distributions, or at least the modes in the case of multimodal distributions. For this reason we look at the performance of ACFTP on the multivariate Normal. A very commonly used method for updating chains is by updating each variable in turn using their conditional distribution, so we use this method here but using the coupling method of Neal (2002) described in Section 4.2.2 to coalesce chains. Each full conditional distribution is Normal and so we generate samples from the full conditionals by scaling and shifting a standard Normal random variable, with each chain using the same standard Normal variable at each iteration. We choose to look at four multivariate Normal distributions

each with mean vector 0 which will be referred to as,

MVN1 A 6 dimensional example with moderate correlation between each variable. This serves as a basic example which should not present any problems.

MVN2 A 6 dimensional highly correlated example. This is the 9 dimensional example used by Neal but with the 3 independent components removed. These 3 components have no effect on the ACFTP procedure as the components will coalesce quicker than and independently of all the other components.

MVN3 A 10 dimensional distribution which has a lot of correlation.

MVN4 A 20 dimensional example with some correlation between each component which provides a higher dimensional example.

The correlation matrices for the first, third and fourth distributions were generated as follows. For an $n \times n$ correlation matrix first an $n \times n$ matrix B of uniform values between -1 and 1 was generated, and then we calculated $A = BB'$. The diagonal of A was then replaced with 1's. This was repeated until the correlations between variables were as desired. The covariance matrix of each distribution was then taken as equal to the correlation matrix. The correlation matrices for each multivariate Normal are given in Appendix C. This method of generating covariance matrices yields multivariate Normal distributions with no pairs of components that are independent of one another, which means that all of these multivariate Normal distributions lead to Gibbs samplers that are slow to mix given their number of dimensions. Distributions arising from real world models tend to have some conditional independence between parameters and so although the multivariate Normal examples are not of very high dimension their high correlation should make up for this.

In the rest of this section we first give recommendations for how to go about choosing the coupling parameters in Neal's (2002) coupling method, and then in Section 4.3.3 go on to show how well ACFTP performs and suggest what values to choose for the parameters required by ACFTP. The robustness of ACFTP to the choice of starting distribution is discussed in Section 4.3.4 and then ACFTP is compared to some currently popular diagnostic methods in Section 4.5.

4.3.2 Choosing the Coupling Parameters

As seen in Section 4.2.2 when implementing Neal's (2002) method for coalescing chains some parameters need to be chosen and these will affect how quickly chains coalesce. Firstly for each component j of the Markov chain a value ω_j, which is half the width

of the Uniform proposals, has to be chosen. For the multivariate Normal examples considered here we use the same value for all components, $\omega_j = \omega$, because the marginal variances of each component are not too dissimilar. Of course we can do this only because we know what the stationary distribution is. In general when approaching a new problem we would use different ω for each component, although it may be convenient to group together subsets of the parameters to use the same ω value and in the examples looked at later this is what we do. Secondly a choice needs to be made about how frequently a Uniform proposal is made in an attempt to coalesce chains. We now look at both of these issues.

Choosing ω

As noted in Section 4.2.5 a disadvantage of Johnson's (1996) coupling method is that choosing the distance between two chains defined to have coalesced requires knowledge of the stationary distribution of the chains. In using Neal's (2002) method it allows us to find a suitable choice of ω using a method that does not require such knowledge, and we show that the acceptance rate of the Uniform proposal can be used as a guide to how quickly chains coalesce.

For each of the four multivariate Normals two chains were started from an over dispersed distribution (taken as a multivariate Normal with mean vector 0 and a covariance matrix of 5 times the identity matrix), and every 20 iterations a Uniform proposal was made in an attempt to coalesce the chains. Various values of ω were tried for each multivariate Normal and the mean of 1000 runs with each ω recorded. We use the mean as a summary of the distribution of the coupling iteration to compare choices of ω simply because the distributions of the coupling iteration tend to have roughly the same shape only with a greater variance as the mean increases. Figure 4.1 shows how the number of iterations taken to coalesce varies with the product of the acceptance rates for each component of the Markov chain. It can be seen from the graphs that a rough guide to choosing a value of ω that leads to reasonably quick coalescence is to choose an ω that gives a probability in the range $(0.2, 0.6)$ of all components of the Markov chain simultaneously accepting their proposals.

Intuitively we would expect the probability to lie away from the extremes of 0 and 1. If the acceptance probability is close to 0 then two chains will expect to wait a long time until they both accept the same state. This corresponds to the situation where the grid of possible proposals is coarsely spread, and although two chains do not need to get very close in order to propose the same state, there will come a point where this benefit does not outweigh the low acceptance probability. Acceptance probabilities near 1 indicate that the grid is too fine, and that it takes a long time for two chains to

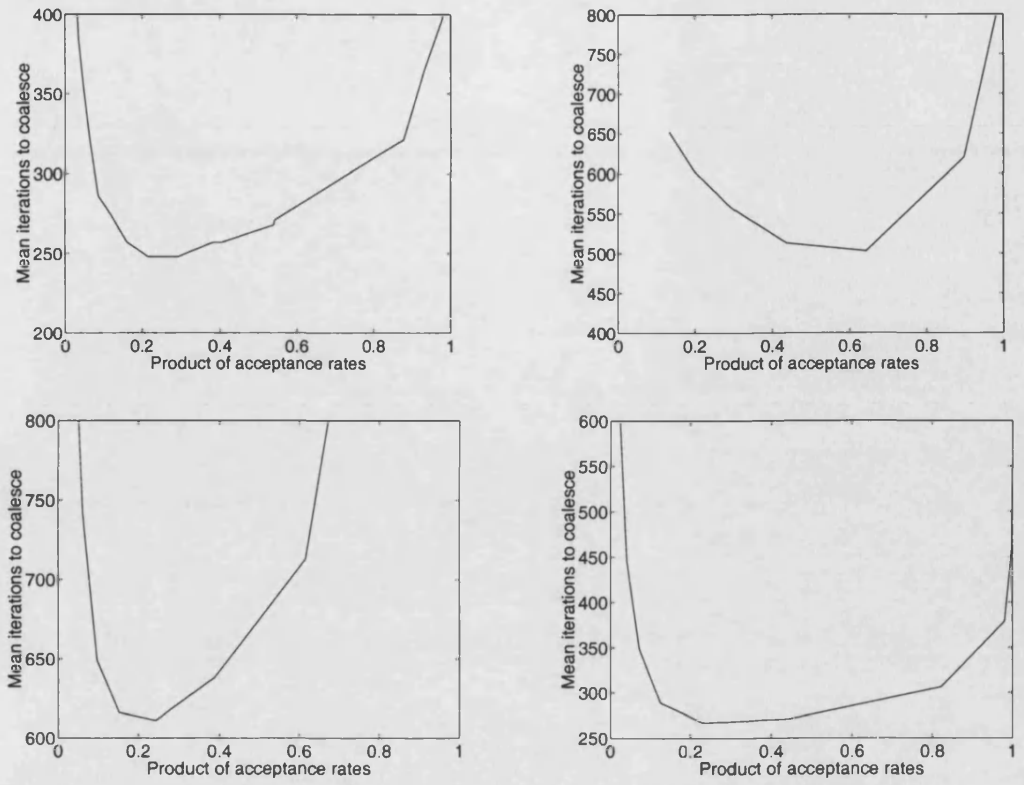

Figure 4.1: Mean coalescence time for various values of ω plotted against the product of the acceptance probabilities. *(top left)* gives results for MVN1, *(top right)* MVN2, *(bottom left)* MVN3 and *(bottom right)* MVN4.

get sufficiently close in order to propose the same state and coalesce. We will see that this rule holds true for the non-multivariate Normal examples considered later.

Finding a suitable value of ω to use for each component does not require a lot of effort, it is easily automated and added to the overall algorithm of ACFTP. Despite this having some security against making a poor choice of ω would be useful. It may also be the case that the distribution under consideration is better suited to several choices of ω, for example if it is multi-modal. We can incorporate these situations into the coalescence step by using different values of ω each time a Uniform proposal is made, by either cycling through different choices of ω or by randomly selecting an ω to use. Some tuning runs will still be required but now they only have to provide a range of ω values within which some quickly coalescing values lie. We now look at whether this has a detrimental effect on the coalescence times. Figure 4.2 gives the mean coupling time for the same 1000 runs used in Figure 4.1 but showing how the coupling iteration varies with the choice of ω. These graphs can then be used to compare the coalescence

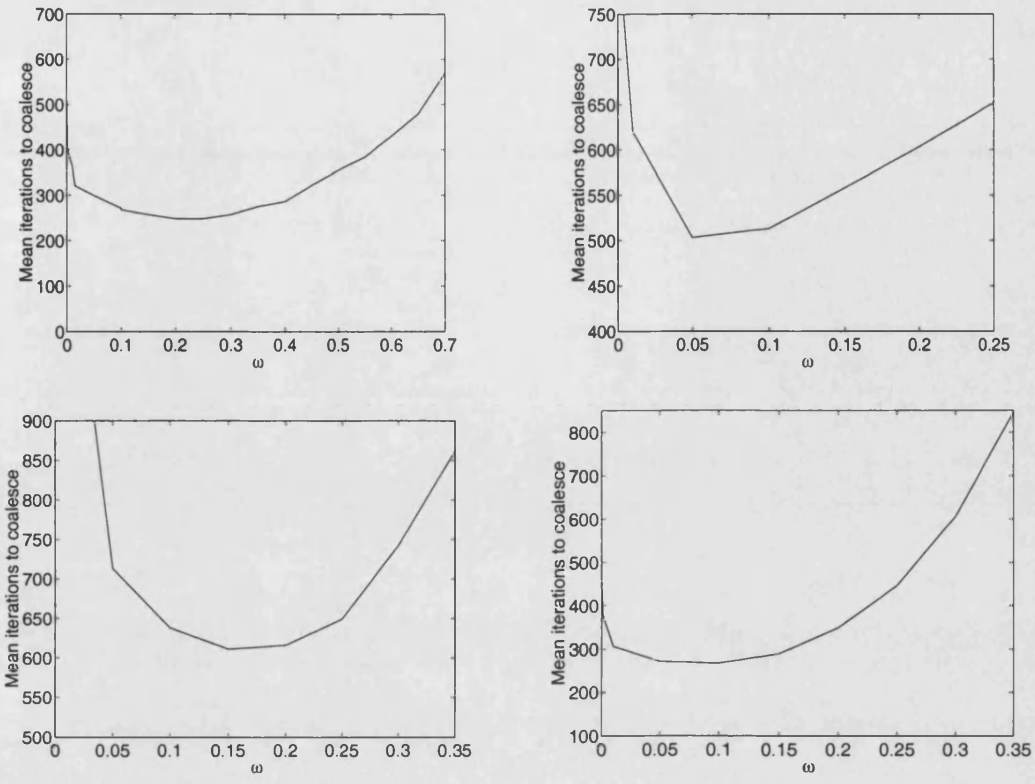

Figure 4.2: The mean coalescence time plotted against ω. *(top left)* gives results for MVN1, *(top right)* MVN2, *(bottom left)* MVN3 and *(bottom right)* MVN4.

times when varying ω, which are now discussed and coalescence results given.

We first consider simply cycling through a fixed sequence of ω values. As an example of a poor choice we look at a scenario where a very wide range of ω values is used and good choices are not used frequently, we consider cycling through choices of ω in the range 0.01 to 1 using 20 equally spaced values in this range including the end values. This is used for all 4 multivariate Normal distributions. Table 4.2 gives the mean coalescence times from 1000 runs for 2 coupled chains using this schedule of ω values, and these compare reasonably well with the coalescence times shown in Figure 4.2 for a single good choice of ω. We would therefore recommend using a range of ω values, believed to include the optimum, where these values of ω result in a product of acceptance probabilities in the range 0.2 to 0.6.

We could instead, with equal probability, choose one of the 20 equally spaced values in the range 0.01 to 1 to use as the value for ω. The results of doing so are also given in Table 4.2 and suggest that this method is not as good as using a fixed cycle of values.

The reason for this comes about through the efficiency of the coalescence step when

multivariate Normal	Mean coupling iteration	
	Fixed cycle	Random
MVN1	360	423
MVN2	563	752
MVN3	782	1094
MVN4	423	2679

Table 4.2: Mean coalescence time for the four multivariate Normal distributions using a fixed cycle of ω values and using a random choice of ω at each iteration.

using poor choices of ω. Using a value of ω that is too large will result in a much worse coalescence time than using a value which is too small by the same relative amount. Figure 4.2 indicates that this is the case. For example, with every multivariate Normal distribution looked at in Figure 4.2 the optimum value of ω to choose is roughly 0.15. If a value of ω three times as large as this were used, then the coalescence time would be much worse than if a value of a third of this were used. The results for large values of ω are not given in Figure 4.2 for clarity because the coalescence times get larger very quickly as ω increases. The quicker coalescence times for the fixed cycle of ω values shown in Table 4.2 come about because the fixed cycle loops through the set of ω values in increasing order, so the smallest ω values appear first. Using the fixed cycle means that on average poor choices of ω values are used after they are by the random scheme. The effect becomes worse for the higher dimensional distributions because at each iteration the ω values are different for each component, and so it is less likely that at a particular iteration every component uses a reasonable choice of ω.

In conclusion we would recommend that a user of ACFTP has a short tuning run where a range of values of ω, which may be different for different components of the chain, are found that result in the product of the acceptance probabilities for each component in the range 0.2 to 0.6. A sequence of quite a few roughly uniformly spaced values in this range should be chosen and these values cycled through at each iteration.

How Often to Try to Coalesce Chains?

We now look at possible schemes for choosing when to use a Uniform proposal in an attempt to coalesce chains. As described in Section 4.2.2 we consider using a Uniform proposal on iterations t where $t \mod L = l$ for values of l in a set $\mathcal{L} \subset \mathbb{Z}$. Figure 4.3 gives the mean coupling iteration (from 1000 runs) for 2 chains simulating from each multivariate Normal distribution using the following four different choices of $\mathcal{L}$, $\mathcal{L}_1 = \{0\}$, $\mathcal{L}_2 = \{0,1\}$, $\mathcal{L}_3 = \{0,1,2\}$ and $\mathcal{L}_4 = \{0,1,2,3\}$. It is clear from these graphs that there is an advantage in having several attempts in a row at coalescing chains, which

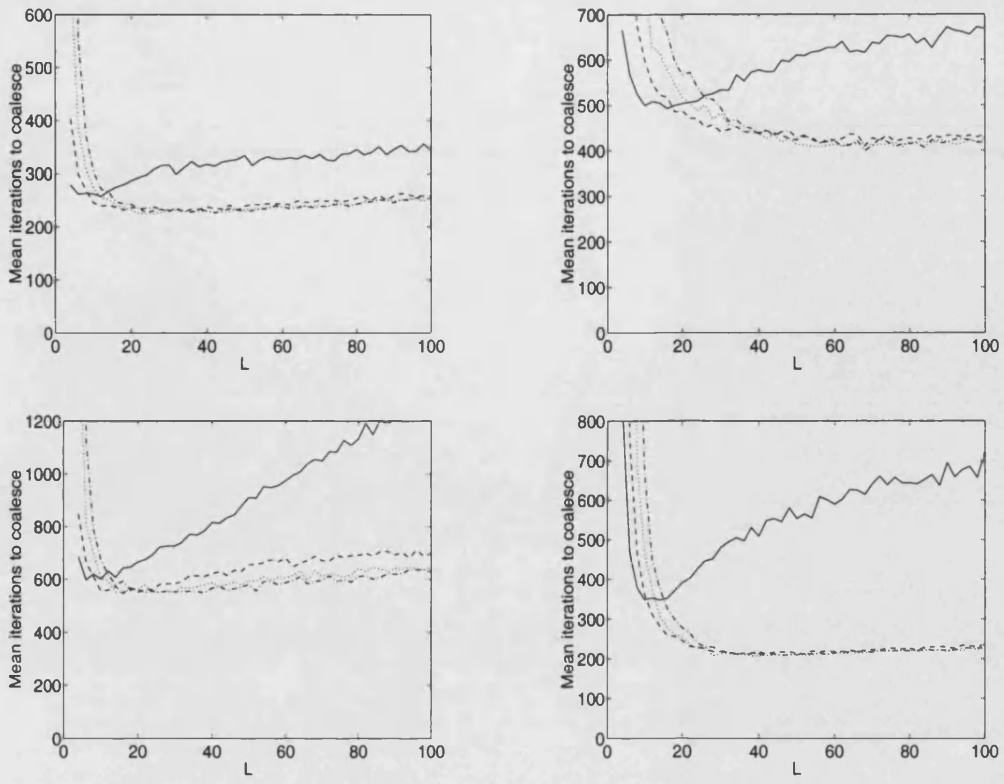

Figure 4.3: Plot of mean coalescence time against L. The solid line indicates when $\mathcal{L}_1$ was used, $\mathcal{L}_2$ a dashed line, $\mathcal{L}_3$ a dotted line and $\mathcal{L}_4$ a dash-dot line. *(top left)* gives results for MVN1, *(top right)* MVN2, *(bottom left)* MVN3 and *(bottom right)* MVN4.

is what happens in the cases $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$. This advantage extends so far that using $L = 100$ with $\mathcal{L}_2$ is better than the seemingly equivalent scheme of using $L = 50$ with $\mathcal{L}_1$, as both schemes do 2 Uniform proposals in every 100 iterations.

Each component of the chain is conditionally dependent on every other component of the chain. For a particular component the acceptance probability in the Metropolis-Hastings step will therefore be dependent on every other component, so for two chains to coalesce all components in both chains need to be similar. If there are a lot of correlated components then getting them all to coalesce at the same time could prove to be unlikely. Having several tries in a row should improve the chances and with each try some components of the chains are coalesced making the two chains even closer at the next iteration. In order to get a product of acceptance probabilities in the range $(0.2, 0.6)$, the acceptance probability for each component has to be high. So as the chains get closer together there is a high chance of getting most of the components to coalesce, but often there will be a few components that do not accept their coalescing

proposals. Having several coalescing proposals in a row serves to collect together the last few components. In the multivariate Normal examples this generally takes only another try. In the examples looked at here, for $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$ on around 50% of the runs the iteration at which chains coalesced was the second Uniform proposal in a row (that is iterations t where $t \mod L = 1$), which suggests why there is little difference between using $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$. For these multivariate Normal examples when including multiple tries at coalescence not only does it generally take less time for two chains to coalesce but the coupling time is more robust to the choice of L.

We would therefore recommend choosing a value of L in the range 20 to 100 with any choice of $\mathcal{L}_i$ for $i = 2, 3, 4$, and from now on when looking at a MVN distribution we use $L = 20$ and $\mathcal{L}_2$. Choices of L larger than 100 have not been shown here, as sensible choices of L lie roughly somewhere in the range of 20 to 100. It is clear that the coupling time will eventually increase as L increases and gets near the coupling time. For problems where chains are very slow to converge then the choice of L can be increased in accordance with a rough estimate of how much longer the chains will take to converge than in the problems seen here, but again the coalescence time should be robust to the choice of L using a sensible choice of $\mathcal{L}$. Results using random choices of L and $\mathcal{L}$ at each iteration are not given here as they perform poorly compared to fixed choices. There are of course many other schemes for choosing when to do a Uniform proposal (and indeed to implement that Uniform proposal), but we feel that the recommended method is robust and efficient. It may be the case that more complicated schemes that are better at coalescing chains, which involve the generation of extra random variables and/or keeping track of extra parameters will take more computation time and then the benefit of a slightly quicker coalescence time is lost.

4.3.3 Choosing the ACFTP Parameters

We now look at the choice of parameters used in ACFTP. That is, m the number of chains started from the starting distribution, r the number of repetitions of chains run from the past, and s the number of samples obtained from the forward runs. To do this ACFTP was repeated 200 times for values of $m = 2, 4, \ldots, 20$, $r = 1, 2, \ldots, 10$ and $s = 1, 2, \ldots, 6$. As previously mentioned the performance of ACFTP can be directly checked by looking at the proportion of times that a chain sampling from the stationary distribution coalesces with the chains used in ACFTP, let $1 - \delta_\pi$ be the estimate of this proportion. For each repetition 20 chains were started independently from the stationary distribution to obtain $1 - \delta_\pi$.

Figure 4.4 gives a summary of the results obtained for ACFTP applied to the 6 dimensional multivariate Normal with moderate correlation. The top left graph plots

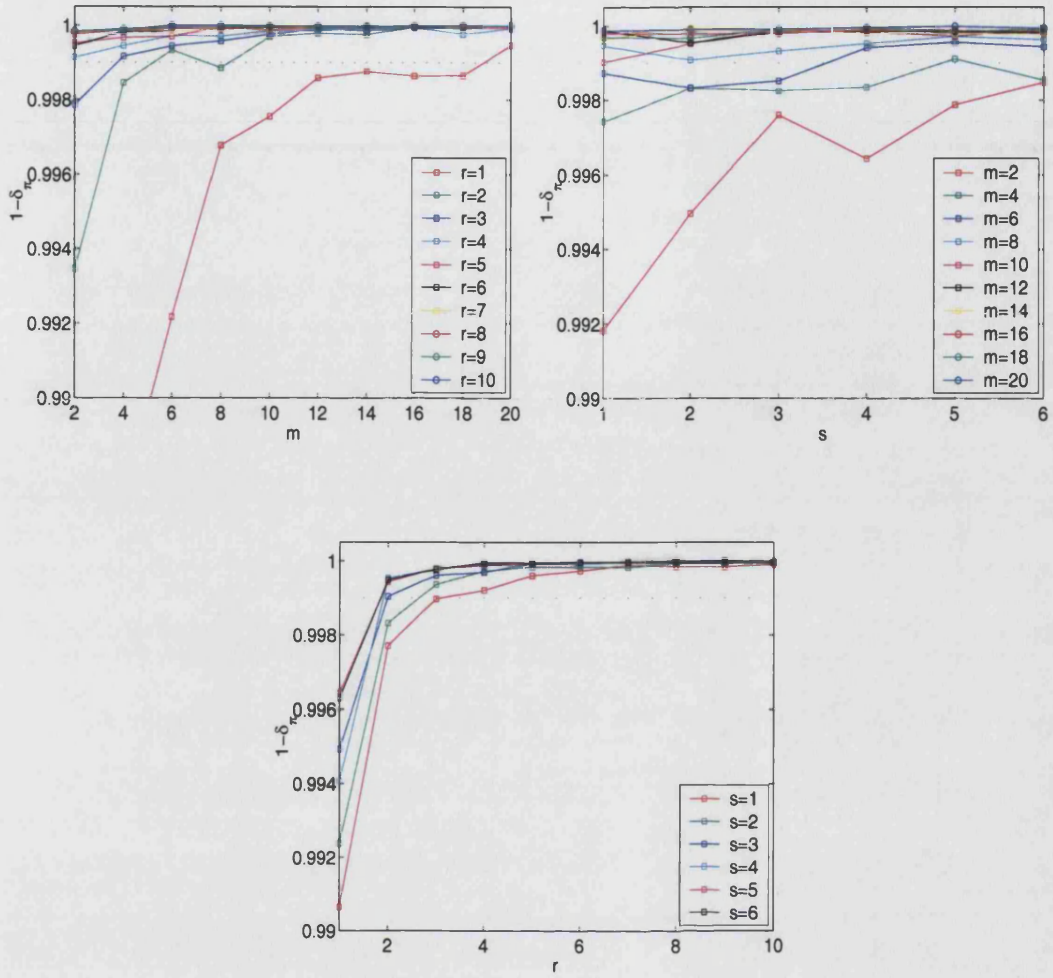

Figure 4.4: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s for MVN1.

the value of $1-\delta_\pi$ averaged over $s = 1, \ldots, 6$, against m for different values of r. The top right and bottom middle graphs plot the average over r against m for different values of s and the average over m against r for different values of s respectively. Similar graphs were produced for the other multivariate Normal distributions and these are shown in Figures D.1, D.2 and D.3.

The dotted line is drawn at $1 - \delta_\pi = 0.999$ as this is felt to be an acceptable level, where according to our argument ACFTP would be returning a sample from the stationary distribution 999 times out of 1000 and on the other occasion is likely to be returning a sample from close to the stationarity distribution if the starting distribution was sensibly chosen. Perfect simulation algorithms could be described as covering 100%

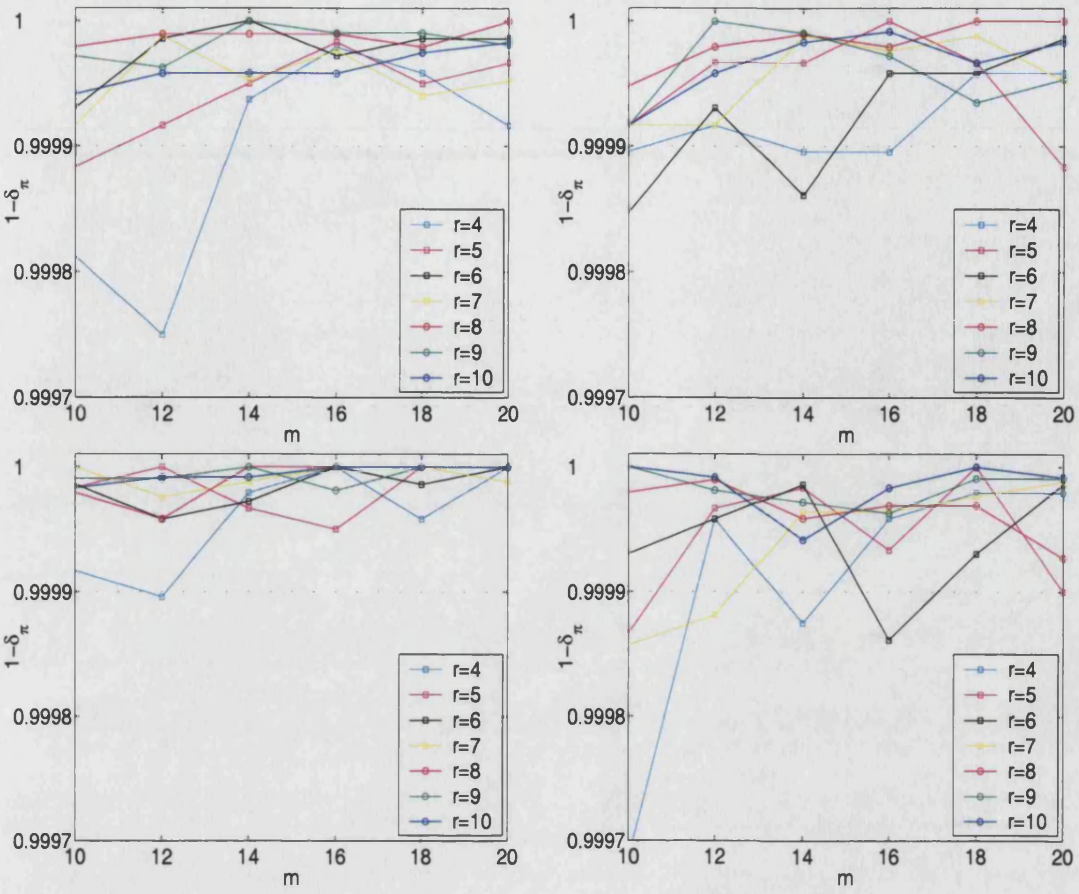

Figure 4.5: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m and r when $s = 3$ for *(top left)* MVN1, *(top right)* MVN2, *(bottom left)* MVN3 and *(bottom right)* MVN4.

of the stationary distribution because the set of states which coalesce to the state at time 0 returned by a perfect simulation algorithm integrates to 1 under the stationary distribution. In choosing 0.999 ACFTP is covering 99.9% of the stationary distribution. We should also consider that ACFTP returns r samples and the chance of getting two or more samples which stationary chains would not coalesce with is less than 0.001.

These graphs suggest that choosing a combination of m, r and s such that $m \geq 10$, $r \geq 4$ and $s \geq 2$ will result in a value of $1 - \delta_\pi$ greater than 0.999. It is clear that the choice of s has less effect especially when using suitably high values of m or r, however the s forward samples provide some reassurance against a poor choice of starting distribution as we will see in the next section. With these choices of m, r and s ACFTP does significantly better than 0.999. Figure 4.5 gives the estimated value of $1 - \delta_\pi$ for recommended choices of m and r with $s = 3$, showing that chains from the

stationary distribution are coalescing on around 0.9999 of occasions.

There are of course different costs involved in choosing r and m. Adding an extra repetition is a good way to improve ACFTP but will be more computationally expensive than adding more chains from the starting distribution. If there are multiple processors available then it is natural to run a repetition on each processor with each extra repetition coming at little extra computational cost, and so we would recommend having as many repetitions as processors. If multiple processors are not available then certainly 2 or 3 repetitions should be used. Increasing m is less computationally expensive and so if $r = 2$ or 3 is used then a value of $m \geq 14$ is recommended.

4.3.4 Sensitivity to the Starting Distribution

We would like to know that ACFTP is robust to sensible choices for the starting distribution. In the previous sections the starting distribution used was simply a multivariate Normal with a covariance matrix of five times the identity matrix, which is the kind of over dispersed distribution envisioned by Gelman & Rubin (1992) that would be good for sampling the initial states of chains from. However we will see that even when using a very bad choice, ACFTP still performs well. The worst possible choice of starting distribution would be a distribution with all its mass on a single point, this will be discussed further in Section 4.7.6. The worst place to put this distribution is at the mode of the multivariate Normal.

Consider a situation where the initial chains are started from a state away from the mode. The first time Step 2 of ACFTP (see Section 3.2) is implemented all chains are started from the same state so they all immediately coalesce. The states returned at time 0 are then very close to the point where all the mass of the starting distribution is placed. If these states are away from the mode of the distribution then the forward runs in Step 3 will drift towards the mode. It is then unlikely in Step 4 that chains started from states near the mode (taken from the forward runs) coalesce to the state away from the mode that the initial chains started from the point mass distribution coalesced to. If however the mass of the starting distribution is close to the mode then it is more likely that chains will stay in the same parts of the state space and coalesce earlier. So we now look at the performance of ACFTP using such a starting distribution.

Figure 4.6 shows the proportion of times that stationary chains coalesce with the ACFTP chains from 200 runs across different values of r and s. All chains start from the same state and therefore coalesce immediately, so a value of $m = 1$ was used. These graphs can be compared to Figures 4.4, D.1, D.2 and D.3 which used a good choice of starting distribution, and it is clear that ACFTP does not do as well with a very poor

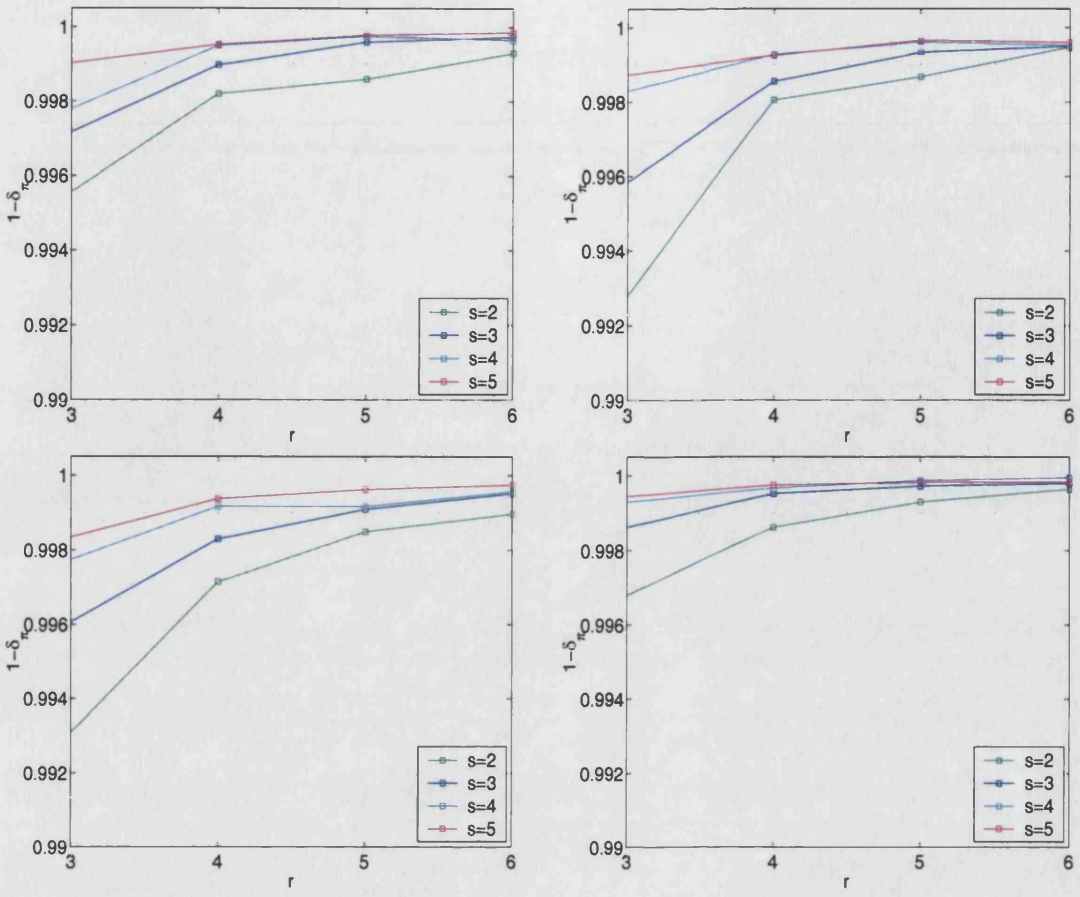

Figure 4.6: Proportion of stationary chains coalescing with ACFTP chains for different combinations of r and s when using a point mass starting distribution, for *(top left)* MVN1, *(top right)* MVN2, *(bottom left)* MVN3 and *(bottom right)* MVN4.

choice of starting distribution. However the proportion of times that the stationary chains coalesce is still close to 0.999. This shows the benefit of using the forward runs to get s states in each repetition to put back into ACFTP, as it is these states that drive the performance of ACFTP here. Of course here there is only one mode and the chains mix reasonably well through the state space. If this were not the case then ACFTP might not do as well and may fail to successfully diagnose convergence. This is however a problem common to all diagnostic methods which can misdiagnose convergence if a chain is poorly mixing. If many chains are used to assess convergence then it can also be incorrectly diagnosed if the starting states for the chains were poorly chosen.

4.4 Applying ACFTP to Other Distributions

4.4.1 A Conjugate Gamma-Poisson Hierarchical Model

Description of Model and Distribution

We now look at how well ACFTP does in assessing convergence of chains simulating from a simple model taken from Murdoch & Green (1998). The data consist of counts of failures x_i, $i = 1, \ldots, N$, in $N = 10$ pump systems at a nuclear power plant and the length of operation time of the pump t_i, $i = 1, \ldots, N$. The number of failures is assumed to follow a Poisson distribution,

$$x_i \sim \text{Poi}(\theta_i t_i) \qquad i = 1, \ldots, N$$

and a conjugate prior is placed on each θ_i so that

$$\theta_i \sim \Gamma(\alpha, \beta) \qquad i = 1, \ldots, N$$

where as in Murdoch & Green (1998) we take $\alpha = 1.802$ and also put a Gamma prior on β,

$$\beta \sim \Gamma(0.01, 1)$$

where $\Gamma(a, b)$ denotes a Gamma density with shape parameter $a > 0$ and scale parameter $b > 0$ and mean a/b. We let $\theta = (\theta_1, \ldots, \theta_N)$. The full conditionals for β and each θ_i are then Gamma distributions,

$$(\beta|\theta, x, t) \sim \Gamma(0.01 + N\alpha, 1 + \sum \theta_i)$$

and

$$(\theta_i|\beta, x, t) \sim \Gamma(1.802 + x_i, \beta + t_i).$$

Implementing ACFTP

In this section we show how the ACFTP procedure performs on this simple example. All the previous examples have involved full conditional distributions consisting of Normal distributions, and these have been coupled by using the same $N(0, 1)$ random variable at each iteration, and then rescaling and shifting it to obtain a sample from the particular Normal distribution sampled from. The same thing can be done with the Gamma distribution. If a random variable X has distribution $\Gamma(a, b)$ then the variable cX has a $\Gamma(a, \frac{b}{c})$ distribution. In each of the variables' conditional distributions the shape parameter does not depend on any other components of the chain so these components

scale	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{2}	10^{3}
mean	20	18	13	12	21	31	36

Table 4.3: Mean coalescence times for choices of ω that are multiples of the scale value.

can be coupled by using the same $\Gamma(a, 1)$ variable at each iteration, (with appropriate fixed value of a), and then rescaling this to suit the scale parameter for a particular chain. Exact coalescence of chains is achieved using Neal's (2002) method in the same manner in which it was used in the previous multivariate Normal examples. For the starting distribution we use the prior, which should be reasonably over-dispersed with respect to the posterior, and we shall later see that this choice of starting distribution is indeed suitable.

Coalescing Chains

Section 4.3.2 gave suggestions on how to go about coalescing chains efficiently using Neal's (2002) coupling method. The first parameters to choose are those defining the widths of the intervals of the Uniform proposals used to coalesce each parameter. For simplicity and because this example is straightforward we will use the same widths for each θ_i, with a different width for β. The general rule of thumb given was to choose these parameters so that the product of the acceptance probabilities across components is in the range $(0.2, 0.6)$.

A quick search for suitable values (using $L = 10$ and $\mathcal{L} = \{0, 1\}$) yields $\omega_\beta = 0.092$ and $\omega_\theta = 0.049$. As a comparison to see how well this choice performs we multiplied these parameters by a factor, given by the scale row in Table 4.3 and looked at the mean number of iterations taken for 2 chains to coalesce using these values. From now on, for convenience we use the term ω to generically denote both the coupling parameter of a component and also the vector of such values, depending on the context. Clearly chains are coalescing very quickly and thus converging rapidly so the results are a little distorted by this, but our initial choice for ω performs reasonably well. Smaller values of ω are coalescing chains more rapidly because the chains get closer together so quickly.

The quick convergence of chains is shown in Figure 4.7 which plots the Euclidean distance between two chains over 10 iterations for 50 runs of chains starting from states sampled from the prior distribution. In almost all runs after 10 iterations each component of the two chains agrees to 3 or 4 decimal places. This means that the two chains have got very close before any attempt is made to coalesce them and so Uniform proposals using small values of ω, and therefore high acceptance rates, will have almost immediate success in coalescing chains, whereas larger values of ω with

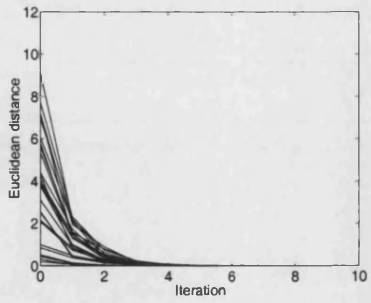

Figure 4.7: Euclidean distance at each iteration between two coupled chains simulating from the pump model. Each line corresponds to one run of two chains.

lower acceptance rates will generally take a few attempts to coalesce chains. In the later examples when chains do not get close so quickly the wider Uniform proposals perform better as they have a chance of coalescing chains when they are still a bit apart. The choice of L is slightly different as well because of the very quick convergence. A smaller value than was previously recommended was chosen, $L = 10$, because it is immediately clear that chains coalesce quickly. Here there is little difference in the choice of $\mathcal{L}$, again because of the very quick convergence, so we took $\mathcal{L} = \{0, 1\}$.

Applying ACFTP

As with the multivariate Normal examples ACFTP was applied to the pump model looking at its performance for different values of m, r and s. Figure 4.8 summarises the results from 200 runs for different values of m, r and s. In each graph the estimated proportion of stationary chains coalescing with the ACFTP chains, $1 - \delta_\pi$, is obtained by taking the average over the parameter not appearing. The graphs are similar to those for the multivariate Normal examples although slightly better. This shows that the recommended choices for m, r and s from Section 4.3.3 perform as well as expected. So from a perfect simulation perspective ACFTP coalesces chains starting from states taken from a set containing more than 0.999 of the stationary distribution.

As with the multivariate Normal distributions ACFTP was run with a single point starting distribution centred at a point near the mode of the posterior distribution. The proportion of stationary chains coalescing is almost the same as that when using the prior as a starting distribution and for any recommended choice of r and s this proportion is greater than 0.9995. So for this example ACFTP is robust to the choice of starting distribution.

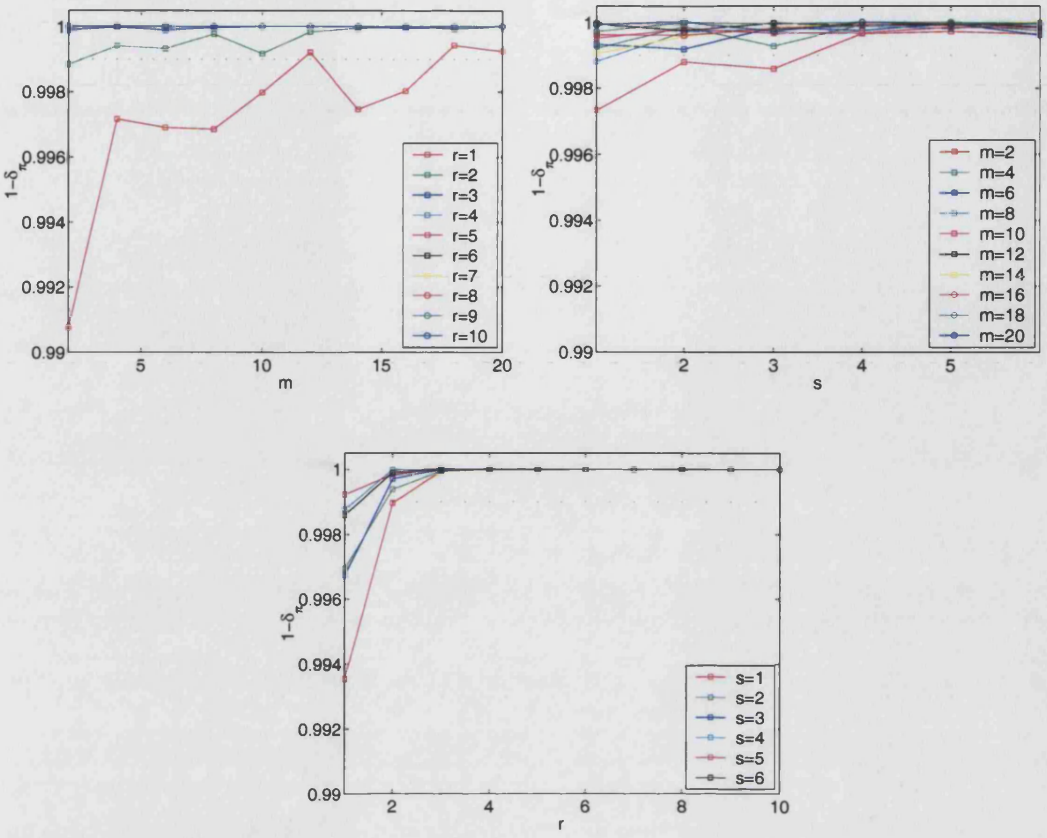

Figure 4.8: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s for the pump model.

4.4.2 A Normal Hierarchical Model

Description of Model and Distribution

In this section we apply ACFTP to an example taken from the BUGS documentation (Spiegelhalter, Thomas, Best & Gilks 1996). The example considers the weights of $N = 30$ young rats whose weight is recorded every 7 days for $T = 5$ weeks. It is assumed that the rats growth curve is linear, the data $x = \{8, 15, 22, 29, 36\}$ consist of the days on which the weights were measured and y_{ij} is the weight of rat i on day x_j

we have the following model.

$$
\begin{aligned}
y_{ij} &\sim N(\alpha_i + \beta_i(x_j - \bar{x}), \tau_c^{-1}) && i = 1, \ldots, N, \quad j = 1, \ldots, T \\
\alpha_i &\sim N(\alpha_c, \tau_\alpha^{-1}) && i = 1, \ldots, N \\
\beta_i &\sim N(\beta_c, \tau_\beta^{-1}) && i = 1, \ldots, N
\end{aligned}
$$

where $\bar{x}$ denotes the means of the x values. The x_js are standardised around their mean to reduce dependence between the α_i and β_is. We use the priors from Spiegelhalter et al. (1996) which are

$$
\begin{aligned}
\alpha_c &\sim N(0, 10^4) & \tau_\alpha &\sim \Gamma(10^{-3}, 10^{-3}) \\
\beta_c &\sim N(0, 10^4) & \tau_\beta &\sim \Gamma(10^{-3}, 10^{-3}) \\
\tau_c &\sim \Gamma(10^{-3}, 10^{-3}),
\end{aligned}
$$

This choice of prior leads to full conditional distributions that are either Normal or Gamma.

Coupling Chains

Chains simulating from this model are easily coupled. Variables with Normal full conditionals are coupled in the same way as the multivariate Normal examples seen in previous sections. Each chain uses the same $N(0, 1)$ variable to generate a sample from a Normal full conditional distribution which at each iteration is scaled and shifted as appropriate. Variables with Gamma full conditionals are coupled as in the Pump example using the same $\Gamma(a, 1)$ random variate at each iteration, for an appropriate fixed value of $a > 0$. Exact coalescence of chains is achieved using Neal's (2002) method in the same manner in which it was used in the previous multivariate Normal examples. In the rest of this section we will show how the methods used in the multivariate Normal examples, and the recommended choices for parameters for both coalescing chains and implementing ACFTP, work on this hierarchical Normal model.

Choice of Starting Distribution

The first thing to consider is the starting distribution. The model comes with what should be a convenient starting distribution, the prior. As discussed in Section 3.4, the prior should form an over dispersed version of the posterior with a reasonable amount of prior weight placed in areas of large posterior probability. For this reason we will use the prior as our starting distribution, although of course we would not recommend taking this approach on a new problem without some initial investigation first. Our

scale	0.001	0.01	0.1	1	10	100	1000
mean	423	434	439	402	419	372	353

Table 4.4: Table showing how the mean number of iterations taken for 2 chains to coalesce (from 1000 runs) varies with the choice of ω. The value in the scale row was multiplied by ω_1 and the mean coalescence time for this new ω vector is given.

results from the multivariate Normal examples suggest that the performance of ACFTP is robust to the choice of the starting distribution and we will see that this is also true for this example.

Coalescing Chains

For each of the parameters α_c, τ_α, β_c, τ_β, τ_c, α_i and β_i for $i = 1, \ldots, N$ a value of ω needs to be chosen. A quick glance at the data suggests that the α_i and β_i will not change drastically with i, so for convenience we use the same value of ω for the α_i and similarly each β_i uses the same value. The recommended method from Section 4.3.2 is to choose ω values that lead to the product of each component's acceptance rate in the range 0.2 to 0.6. For the rats model with 65 parameters a small number of short tuning runs (around 6) using $L = 20$ and $\mathcal{L} = \{0\}$ were needed to get acceptance rates for each parameter in the range $(0.2^{1/65}, 0.6^{1/65})$, where for convenience we measured the overall acceptance rates for the α_i and β_i's and assumed that the acceptance rate did not vary much over the value of i. Let ω_1 denote the vector of Uniform proposal widths chosen using our tuning runs. Table 4.4 shows that this choice performs well compared to other choices. In this example there is not much variation in the coalescence times across different choices of ω and the coalescence time is very robust to the choice of ω, so any choice based around the originally chosen values would work equally well. So for the rest of this section we used 20 values of ω in the range $0.1\omega_1$ to $10\omega_1$.

We have also looked at the effect the choice of L and $\mathcal{L}$ has on the coalescence times. We would expect the choice of $\mathcal{L}$ to matter less here than in the multivariate Normal examples because there is quite a bit of conditional independence between the variables in the model. For example each set of parameters (α_i, β_i) for $i = 1, \ldots, N$ is conditionally independent. Figure 4.9 gives the mean coalescence time from 1000 runs of two coupled chains using different values of L and $\mathcal{L}$ where $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{L}_3$ and $\mathcal{L}_4$ are defined as in Section 4.3.2. Here the choice of $\mathcal{L}_1$ does not perform as badly for large values of L as it did in the multivariate Normal examples as expected. Any value of L in the range suggested in Section 4.3.2 performs equally well, as shown in Figure 4.9. Foe values of L larger than shown on the graphs, the coupling time slowly increases as

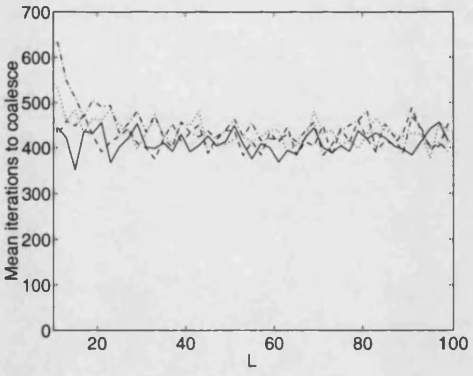

Figure 4.9: Plot of L versus mean coalescence time in the rats model. The solid line indicates when $\mathcal{L}_1$ was used, $\mathcal{L}_2$ a dashed line, $\mathcal{L}_3$ a dotted line and $\mathcal{L}_4$ a dash-dot line.

L gets larger. So in the rest of this section we use $L = 30$ and $\mathcal{L}_2$.

In this example following the recommended approach to choosing the parameters used to coalesce chains given in Section 4.3.2 would lead to good choices for ω, L and $\mathcal{L}$ that coalesce chains reasonably quickly compared to an optimal choice which would require a lot more effort to find. From Table 4.4 we can see that using an optimal value for ω should coalesce two chains in around 350 iterations whereas Figure 4.9 shows that our choice takes about 430.

Performance of ACFTP

As in the previous examples, for different values of m, r and s ACFTP was used to diagnose convergence, using the prescribed coalescing method and starting distribution. For all combinations of $m \geq 6$, $r \geq 3$ and $s \geq 2$ the estimated value of $1 - \delta_\pi$ was greater than 0.9995, showing that in this example ACFTP is performing as desired. Generally the results followed the same pattern as those for the multivariate Normal distributions and the pump model, with a choice of $m \geq 10$, $r \geq 4$ and $s \geq 3$ leading to a proportion greater than 0.9999 of stationary chains coalescing. ACFTP was also run using a single point starting distribution with its mass near the mode of the distribution. The performance of ACFTP followed the same pattern as in previous examples, with a slightly lower proportion of stationary chains coalescing to the returned state at time 0 than with an over-dispersed starting distribution. However using recommended values for m, r and s should still lead to a value of $1 - \delta_\pi$ above 0.9995.

4.5 Comparing ACFTP to Diagnostic Methods

Comparing diagnostic methods is not a straightforward issue as there are many factors that could be considered when assessing how good a method is at diagnosing convergence. For example we could consider how well it correctly diagnoses convergence, how convenient it is to use, and how computationally demanding the method is. We have already looked at how well ACFTP diagnoses convergence by estimating $1-\delta_\pi$ and seen that this proportion should be near 1 if ACFTP is used in the recommended manner. The main usability requirement is that problem specific programming is not required (beyond that needed to run the chain). Despite appearing to be the case, ACFTP does not require any problem specific programming. The routines for running chains from the past, keeping track of seeds and running chains forward to obtain samples can be written as free standing code. The function that updates the state of a chain from one iteration to the next simply needs to take the required seeds as input. For ACFTP the computation time is clear, it's the time taken to return the samples at time 0, but for the other diagnostics it is less clear, because as well as the time taken by the computer to calculate the diagnostics, there is also the time taken and effort involved in interpreting the diagnostic output which is not an automated process.

In this section we will look at a few of the most commonly used diagnostics, which were briefly described in Section 1.5. The four diagnostics we look at are those popularised by CODA (Best et al. 1995), which are due to Gelman & Rubin (1992), Raftery & Lewis (1992*a*), Geweke (1992) and Heidelberger & Welch (1983).

As well as proposing different diagnostics, different approaches to overcome the uncertainty in the diagnostics have been considered. The main discussion of this has been over the advantages and disadvantages of one long run over multiple shorter runs, (Raftery & Lewis 1992*b*, Gelman & Rubin 1992). The argument for one long run is that the distribution of the samples from the end of a long run will be closer to the stationary distribution than any of the samples from short runs. However the multiple runs guard against individual chains getting stuck and not exploring the entire state space (or at least that part that is interesting), and the uncertainty in this is something that cannot be measured. So it is widely advocated to use a few multiple chains, see for example Cowles & Carlin (1996).

These diagnostics were applied to the example distributions used in this chapter, that is the four multivariate Normal distributions, the rats data model and the pump data model. We applied the diagnostics, in a way that might reflect how a user would go about diagnosing convergence, to five chains starting in states sampled from the same over-dispersed starting distribution as that used by ACFTP. Each diagnostic

was applied to iterations 1-2000, 2001-4000, 4001-8000 and 8001-16000 as if the user was following a procedure of running the chains for an initial time and then if a lack of convergence is diagnosed running the chains for as long as they had already been run. The implementation and interpretation of each diagnostic varies greatly and the diagnostics do not come with precisely stated rules as to how to apply them. Issues that arise when using diagnostics include, how long to further run a chain if convergence has not been diagnosed in the current run, which portion of the output to apply the diagnostics to, and exactly how to detect convergence. Some diagnostics also have parameters which must be chosen, for example the quantile of interest in Raftery & Lewis's (1992*a*) method. However since we are only roughly comparing the number of iterations at which each diagnostic suggests convergence we do not look at their implementation in detail, but we do now describe the methods used in this thesis when applying each diagnostic.

Convergence using Gelman & Rubin's (1992) diagnostic is concluded if the upper 95% quantile of the CSRF (Brooks & Gelman 1998) is less than 1.1 for all components of the chain. Various ways of assessing convergence using the Gelman & Rubin (1992) method are given in the literature, sometimes a value of 1.2 is given but using 1.1 is more in line with the recommendations in the BOA manual from which we are taking our default settings for the diagnostics. We have also looked at the multivariate potential scale reduction factor (MPSRF) (Brooks & Gelman 1998) which is a multivariate version of the PSRF of Gelman & Rubin (1992). This should be judged on the same scale as the univariate scale reduction factors.

For each specified quantile Raftery & Lewis's (1992*a*) diagnostic automatically returns a number $N_{\min}$ for each parameter of the chain which is the number of iterations to discard from the beginning of the run, using the default quantile of 0.95 we take the maximum value of these as the burn-in time. The default accuracy given by BOA was used which is 0.005.

Geweke's (1992) method for diagnosing convergence is less specific in when to consider convergence has been reached. A statistic is returned for each component of the chain that should converge to the standard Normal distribution as the chain converges to stationarity, but the assessment of this is left to the user. For the purposes of this thesis this assessment was done by rejecting convergence if at least two components had values greater than 2.5, except for the rats example where because of the larger number of parameters a value of 3 was used. Frequently the statistics exceeded 5 making rejection of convergence straightforward. The default proportion of iterations from the beginning and end of the run specified in BOA were used to calculate the convergence statistics, which were 0.1 and 0.5 respectively.

	MVN1			MVN2			MVN3			MVN4		
t	G	HW	Both	G	HW	Both	G	HW	Both	G	HW	Both
2000	0	0	0	0	0	0	0	0	0	0	0	0
4000	1	2	0	0	1	0	0	0	0	3	1	0
8000	2	2	1	1	2	0	1	0	0	1	4	0
16000	3	0	0	1	1	0	3	0	0	3	3	2

Table 4.5: The number of chains out of five assessed to have converged by iteration t for the four MVN distributions, using Geweke's (G), and, Heidelberger and Welch's (HW) diagnostics. The column headed Both shows how many of the chains both Geweke's and Heidelberger and Welch's diagnostics suggest convergence.

Heidelberger & Welch's (1983) diagnostic returns a pass or fail as to whether the chain has reached stationarity or not and, if it has, gives the number of iterations to discard from the beginning of the run. If all the parameters of the chain pass the test then the maximum number of iterations taken for a component to converge is taken as the burn-in time.

Multivariate Normal

For the four multivariate Normal distributions introduced in Section 4.3.1 assessment of convergence varied considerably across distributions and diagnostic methods. For all four multivariate Normal distributions both Brooks & Gelman's (1998) MPSRF and Gelman & Rubin's (1992) methods conclude convergence within 2000 iterations, and Raftery & Lewis's (1992*a*) method diagnoses convergence after about 100 iterations.

Geweke's (1992) and Heidelberger & Welch's (1983) diagnostics are much less optimistic about convergence. Table 4.5 gives a summary of when these two methods diagnose convergence for each MVN. Heidelberger & Welch's (1983) method suggests convergence for most of the chains in the 20 dimensional distribution by 8000 iterations and recommends a burn-in of around 4000, the chain failing the test had only 1 parameter fail. For the other 3 distributions a user of this diagnostic would definitely continue to burn-in chains after 16000 iterations. An optimistic user of Geweke's (1992) method, as the interpretation of statistics is at the discretion of the user, would probably conclude convergence after 16000 iterations for all the examples, so a burn-in of 8000, except the highly correlated 6 dimensional example for which the user would run chains considerably after 16000 iterations. It is the high autocorrelations in some of the chains that lead to Geweke's (1992) and Heidelberger & Welch's (1983) methods not diagnosing convergence. For some components of the 10 dimensional MVN the estimated lag 50 autocorrelations are still around 0.7, while both 6 dimensional

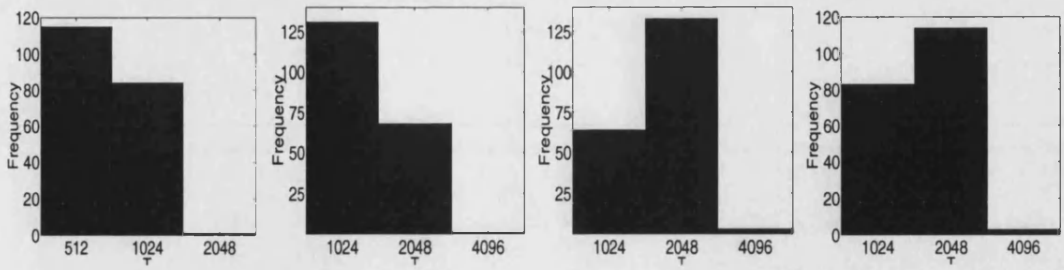

Figure 4.10: Histograms of the number of iterations in the past that ACFTP went back to in 200 runs. From left to right the graphs correspond to MVN1, MVN2, MVN3 and MVN4.

multivariate Normals have some estimated lag 50 autocorrelations above 0.2.

We can compare these results with the furthest time in the past that ACFTP goes back to. Results of the time in the past that ACFTP went back to, from 200 runs, with $m = 10$, $r = 5$ and $s = 3$ are shown in Figure 4.10. We can see that this roughly agrees with Gelman & Rubin's (1992) diagnostic, which diagnosed convergence in a couple of thousand iterations, with the lower correlation examples generally having to be run from less far in the past. Using this choice of ACFTP parameters resulted in greater than 0.9995 of chains started from the stationary distribution coalescing with the ACFTP chains in all 4 examples, demonstrating that discarding the first 4000 iterations of each chain is easily sufficient. Following the general recommendations of using multiple diagnostics then for these examples chains would be run for a lot longer than necessary.

Another consideration is how long the methods take to assess convergence. ACFTP takes around 10-15 seconds[1] to complete for each of the MVN examples, and at the end of this the program can automatically generate the desired number of simulations without out any extra user interaction. With the other diagnostics the time considerations are less well defined because the user must interpret numerical or graphical output. The calculation of diagnostics is not necessarily quick on its own either. For the 20 dimensional example running Geweke's (1992) diagnostic on 2000 iterations of 5 chains takes around 15 seconds[2] while Heidelberger & Welch's (1983) method takes around 90 seconds to output its results. Gelman & Rubin's (1992) and Raftery & Lewis's (1992*a*) methods take a couple of seconds. Therefore including in the extra time taken to interpret results, ACFTP is just as time efficient as the other diagnostics considered here.

[1] All times relating to ACFTP were obtained by running C code on one 400MHz UltraSPARC II processor.

[2] All times relating to producing diagnostic results were obtained using the R version of BOA running on a 400MHz UltraSPARC II processor.

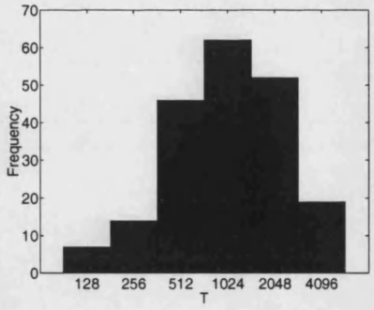

Figure 4.11: Histogram of the number of iterations that ACFTP goes back to for 200 runs in the rats model.

The Pump and Rats Examples

The pump model is very straightforward with all diagnostics indicating convergence in under 100 iterations or less. On almost all runs ACFTP ran chains from 32 iterations in the past. Both running ACFTP and producing diagnostics took a few seconds.

For the rats model Gelman & Rubin's (1992) method diagnoses convergence in around 300 iterations, while Raftery & Lewis's (1992*a*) methods suggests convergence in a few hundred iterations. Heidelberger & Welch's (1983) and Geweke's (1992) diagnostics indicates that convergence has been reached between 500 and 1000 iterations depending on the chain. However the MPSRF (Brooks & Gelman 1998) disagrees with these and is still 1.32 after 8000 iterations. Brooks & Gelman (1998) experience the same thing in their paper when they look at a model with 141 parameters where the individual PSRFs indicate convergence but the MPSRF does not. Brooks & Gelman (1998) do not believe that this is a result of the high number of dimensions but rather an indication of lack of convergence for functions of the parameters other than the individual values themselves. Figure 4.11 shows the times in the past that 200 runs of ACFTP went back to using $m = 10$, $r = 5$ and $s = 3$. It shows that ACFTP sometimes runs the chains for a little longer as a burn-in but not substantially so and generally agrees with the other diagnostics. The time taken to calculate the diagnostics is a few seconds for Gelman & Rubin's (1992) methods, around 20-30 seconds to get Geweke's (1992) and Raftery & Lewis's (1992*a*) diagnostics and about 2 minutes for Heidelberger & Welch's (1983), while ACFTP takes on average around 30 seconds to complete a run.

So for both of these examples, in terms of time taken, ACFTP takes no longer than using a selection of diagnostic methods once the time taken to interpret the diagnostics output is included.

4.6 Comparing ACFTP to Perfect Simulation Algorithms

We can also consider how well ACFTP compares to perfect simulation algorithms. Reasonably efficient perfect simulation algorithms are currently not available for most distributions arising from Bayesian models, those that do exist can only be applied to distributions with certain properties. Of the Bayesian examples looked at in this thesis only the pump model is sufficiently tractable.

Murdoch & Green (1998) give two methods for obtaining a perfect sample from the pump model. The first method involves truncating the distribution to a bounded state space and then finding bounds on the conditional distributions within subsets of the state space that form a finite partition. This is not really generating a perfect sample from the specified problem, but generating one from a truncated version. The idea of truncating the state space to a bounded region is one that has also been used by Philippe & Robert (2001) and Guglielmi et al. (2001). Bounding the state space means that finding a perfect simulation algorithm may be easier. In truncating the state space the same idea which is behind ACFTP is used, getting chains starting from states taken from a set which has total probability close to 1 under π coalescing to the same state at time 0. The second of Murdoch & Green's (1998) methods uses scaling properties of the Gamma distribution to remove this restriction and create a perfect sampler on the desired distribution rather than a distribution arising from using a truncated prior. Results given in Murdoch & Green (1998) show that their algorithms are computationally quick. However there is a lot of problem specific algebraic work to do before either form of their perfect sampler can be implemented. ACFTP requires no algebraic effort to implement. The results from Section 4.4.1 show that we can consider ACFTP to be returning a sample from the stationary distribution on around 0.9999 of occasions using the recommended number of chains. ACFTP could be considered as a perfectly viable alternative to perfect simulation in this case.

All of the current perfect simulation algorithms cannot sample from the rats model (at least in a usable way), and so in this case ACFTP improves on any perfect simulation algorithms.

4.7 Variable Dimension Problems

4.7.1 Introduction

There are some situations in statistical inference where the dimension of the parameters in the model vary. An example of this type of problem is looking at data that come from a number of different populations, but where the number of populations is

unknown; these are called mixture models and are looked at in Section 4.7.5. Green (1995) introduced a flexible method for updating a Markov chain simulating from a distribution where the dimension of the parameter vector is not fixed. The algorithm of Green (1995) allows a prior to be placed on the unknown number of elements of the model, with further priors put on the variables in each element, so that the data can be modelled in a fully Bayesian manner. Green (1995) showed how to construct proposals and the correct acceptance probabilities to allow chains to jump between different dimensions and maintain detailed balance with respect to the specified distribution.

More recently Stephens (2000) introduced the idea of viewing a Markov chain simulating from a variable dimension problem as a marked point process and allowing the chain to move through different dimensions via a birth and death process. The methods of Green (1995) and Stephens (2000) are described in Sections 4.7.2 and 4.7.3 respectively.

It should be noted that by no means are these the only ways to investigate models where the dimension of the parameter space is unknown. Several Markov chain can be used, each simulating from a different model with a fixed number of parameters (e.g. a fixed number of populations in a mixture), and the results from the separate chains compared. Carlin & Louis (1996) and Green (2003) describe how two models can be compared by looking at the probability of the data given the first model divided by the probability of the data given the second model, this ratio being the Bayes factor.

Unfortunately the wealth of convergence diagnostics for fixed dimension chains (Cowles & Carlin (1996) and Brooks & Roberts (1998)) cannot be directly applied to ones whose dimension varies. Richardson & Green (1997) and more recently Castelloe & Zimmerman (2002) and Brooks & Giudici (1999) have suggested methods for diagnosing convergence in variable dimension chains, there are however significant drawbacks to these methods, which are described in Section 4.7.4. The approach of ACFTP can be applied in variable dimension problems. The requirement is the same as that needed for ACFTP to work with fixed dimension problems; coupled chains need to coalesce. The parameter which determines the number of dimensions is just an extra parameter which must be able to coalesce. We apply ACFTP to a couple of mixture models; a mixture of an unknown number of Normal densities (Richardson & Green 1997) and a mixture of regression lines (Hurn, Justel & Robert 2003).

4.7.2 Green's Reversible Jump MCMC

Green (1995) considered the situation where there are a countable number of candidate models indexed by k each with a parameter vector θ, whose dimension varies over k. A model for the data y is defined hierarchically by first defining a prior probability for each

model, $p(k)$, and then a prior on the parameters in each model, $p(\theta|k)$. The structure of the data given a particular choice of model is defined through the likelihood $l(y|\theta,k)$. Combining the prior and likelihood leads to a posterior distribution π over states of $x=(k,\theta)$. Green (1995) constructed a Metropolis-Hastings step which simulates from π as follows.

Let there be a countable number of possible move types indexed by $m=1,2,\ldots$ which are used to update the chain. If the current state of the chain is x, at each iteration a move is proposed of type m, and a new state x' is proposed. These are generated from the joint probability measure $q_m(\cdot,\cdot)$. The move to x' is then accepted with probability,

$$\alpha_m(x,x') = \min\left\{1, \frac{\pi(dx')}{\pi(dx)}\frac{q_m(x',dx)}{q_m(x,dx')}\right\}$$

where the ratio is taken with respect to a suitable dominating measure, the existence of which is ensured by the dimension matching condition placed on the proposals. This is now made clearer by considering a simple example.

Let $p(x|y)=p(k)p(\theta|k)l(y|\theta,k)$, which is proportional to the posterior distribution. Suppose for example that a move of type m is chosen with probability $r_m(x)$ that proposes a move from state $x\in\mathbb{R}^{n_1}$ to $x'\in\mathbb{R}^{n_2}$. This proposal can be made by first generating a vector of random variables u with dimension m_1 from the density $q_1(\cdot)$ and setting x' to be a function of x and u. The reverse move, using a vector of random variables u' of dimension m_2 generated from a density $q_2(\cdot)$, is constructed so that a bijection between (x,u) and (x',u') exists, in particular it must be the case that $n_1+m_1=n_2+m_2$. The state x' is then accepted as the new state of the chain with probability,

$$\min\left\{1\ ,\ \frac{p(x'|y)}{p(x|y)}\frac{r_m(x')q_2(u')}{r_m(x)q_1(u)}\left|\frac{\partial(x',u')}{\partial(x,u)}\right|\right\} \tag{4.2}$$

where $|\cdot|$ denotes the Jacobian arising from the change of variables from (x,u) to (x',u').

4.7.3 Stephen's Birth and Death Process

When considering mixture distributions Stephens (2000) proposed a method for moving between different dimensions based on a birth/death process. Let $x=(k,w,\phi,\eta)$ denote the state variable, where k is the number of components in the mixture distribution, $w=(w_1,\ldots,w_k)$ contains the weights of each component that sum to 1, $\phi=(\phi_1,\ldots,\phi_k)$ contains the vector of parameters for each of the k components and η is a vector of parameters common to all components. Stephens (2000) considers

posterior distributions of the form,

$$p(k, w, \phi, \eta|y) \propto p(\eta)p(k|\eta)p(w|k, \eta)p(\phi|k, w, \eta)L(k, w, \phi, \eta; y) \tag{4.3}$$

where the first four terms make up the prior and the last term is the likelihood of the data y,

$$L(k, w, \phi, \eta; y) = \prod_{i=1}^{n} \{w_1 f(\phi_1, \eta; y_i) + \cdots + w_k f(\phi_k, \eta; y_i)\}.$$

For a fixed value of η Stephens (2000) notes that the distribution $p(k, w, \phi|\eta, y)$ can be viewed as a marked point process with each point ϕ_j having an associated mark w_j. We now describe the method of Stephens (2000) for constructing a Markov chain with a stationary distribution defined by (4.3). The method we give here uses the prior to generate proposed new components, this was used successfully by Hurn et al. (2003) from where we take our example and so is the method used in this thesis.

Let the current state of the chain be $x = (k, w, \phi, \eta)$. For a fixed time t_0 run the following birth-death process with constant birth rate $\beta(x)$.

1. Calculate the death rate for each component,

$$\delta_j(x) = \beta(x)\frac{L(x\backslash(w_j, \phi_j); y)}{L(x; y)}\frac{p(k-1)}{kp(k)}$$

   where $x\backslash(w_j, \phi_j)$ is the state x but with the j^{th} component removed and the weights reweighted to sum to one.

2. Calculate the total death rate $\delta(x) = \sum_j \delta_j(x)$.

3. Generate the time to the next jump from an exponential distribution with mean $1/(\beta(x) + \delta(x))$.

4. Choose the type of jump with probabilities,

$$\mathbb{P}(\text{birth}) = \frac{\beta(x)}{\beta(x) + \delta(x)} \quad \text{and} \quad \mathbb{P}(\text{death}) = \frac{\delta(x)}{\beta(x) + \delta(x)}.$$

5. If a birth is chosen increase k by 1, simulate a new component for w_k from $p(w_k|k, \eta)$ (re-weighting the w_j so that they sum to one) and a new component ϕ_k from $p(\phi_k|w, k, \eta)$. If a death is chosen, then remove component j with probability $\delta_j(x)/\delta(x)$ and decrease k by 1 after relabelling the components, and re-weighting the w_j so that they sum to one.

After time t_0 update w, ϕ and η as for the fixed k case using whatever type of sampling method desired.

4.7.4 Problems of Diagnosing Convergence

The large number of convergence diagnostics for fixed dimension chains (see Section 1.5) cannot be directly applied to chains whose state varies in dimension. These diagnostics cannot be applied to parameters particular to a model (for example parameters describing a component in a mixture) because the meaning and number of them changes when the model changes. Two approaches have been suggested for this problem: firstly to monitor convergence of the model parameter (for example the number of components in the mixture) and then when this has been reached, to assess convergence within each of the models visited; secondly monitoring convergence of the model parameters that retain the same meaning across different models.

Brooks (1997) suggests the former but acknowledges that even in very long runs there will be infrequent visits to some models and that convergence within these models will almost certainly not have been reached. The problem of assessing convergence then becomes one of deciding which models to monitor for convergence and this decision has to be made without recourse to any diagnostics.

Castelloe & Zimmerman (2002) and Brooks & Giudici (1999) suggest the latter and take a Gelman & Rubin (1992) style approach of running multiple chains, and splitting up the total variation of functions of the parameters common to every model, to produce statistics that indicate convergence. As with many diagnostics applied to output from a chain Castelloe & Zimmerman (2002) and Brooks & Giudici (1999) suggest different schemes for choosing the parts of the output to apply their diagnostics to. In the following descriptions of their methods we do not discuss this.

For the parameters with the same interpretation across models θ, Brooks & Giudici (1999) suggest monitoring a univariate summary of θ, $\bar{\theta}$. Using the output from parallel chains Brooks & Giudici (1999) split the total variation of $\bar{\theta}$ into between and within model variances. They suggest that chains have converged when the estimates of the variances have converged to the same value.

Castelloe & Zimmerman (2002) take a multivariate approach similar to Brooks & Gelman's (1998) multivariate version of Gelman & Rubin's (1992) diagnostic and consider cases where θ has more than one dimension. Multivariate versions of the estimates of between chain, within chain and total variance are calculated, from which multivariate scale reduction factors are produced. Convergence of the chains is said to have been reached when these scale reduction factors are close to 1, and the eigenvalues of certain functions of the variance matrices have converged. Castelloe & Zimmer-

man (2002) also suggest optionally looking at the univariate estimates of the variance components and scale reduction factors of components of θ. In order to increase the number of parameters monitored they suggest including some allocation parameters in θ, that is monitoring which component in a mixture an observation is modelled as coming from. For large datasets following the allocation of every observation will be overly demanding so they suggest choosing a few which might be important.

Both of these methods suffer from the fact that they are using only the subset of parameters with the same interpretation across models. It is clearly possible that these parameters could converge before the chain as a whole and on such occasions the methods of Brooks & Giudici (1999) and Castelloe & Zimmerman (2002) would prematurely diagnose convergence. ACFTP on the other hand does not have this problem because all the parameters of two chains must be the same before they coalesce providing a diagnostic based on a chain converging as a whole.

4.7.5 Mixture Models

The example looked at in the next section is taken from Richardson & Green (1997) and based on the following model structure. Each observation y_i is independent and modelled as

$$y_i \sim \sum_{j=1}^{k} w_j f_j(\cdot|\theta_j) \qquad i = 1, \ldots, n$$

where each $f_j(\cdot|\theta_j)$ is a density with parameter vector θ_j and the component weights are w_j which sum to 1. We assume that this model arises from the situation where there are k populations, the number of which is unknown, and that each of the observations is drawn from one of the populations, j, with probability w_j in a manner described by the density $f_j(\cdot|\theta_j)$. To aid simulation from this model auxiliary variables $z = (z_1, \ldots, z_n)$ are introduced where $z_i = j$ if observation i is modelled as coming from population j. Priors are placed on the parameters (w, θ, k), where $w = (w_1, \ldots, w_k)$ and $\theta = (\theta_1, \ldots, \theta_k)$. It is of course possible to include an extra layer of priors with hyperparameters, which will be seen when we look at the particular model from Richardson & Green (1997).

4.7.6 A Mixture of Univariate Normals

The Model

Richardson & Green (1997) looked at data consisting of the speeds of 82 galaxies relative to our own. They considered that the galaxies came in clusters where the speeds within each cluster could be modelled as coming from a Normal distribution, but where the

number of clusters is unknown. We apply ACFTP to the same model as used by them. Each parameter vector $\theta_j = (\mu_j, \sigma_j^2)$ consists of the mean μ_j and variance σ_j^2 of each Normal distribution in the mixture, so that

$$f_j(y|\theta_j) = f_j(y|\mu_j, \sigma_j^2) = \frac{1}{\sigma_j\sqrt{2\pi}} \exp\left\{-\frac{(y-\mu_j)^2}{2\sigma_j^2}\right\}.$$

Richardson & Green (1997) use a prior structure which has each $\mu_j \sim N(\xi, \kappa^{-1})$ and $\sigma_j^{-2} \sim \Gamma(\alpha, \beta)$. In following the example of Richardson & Green (1997) we also impose the same ordering constraint on the vector μ by requiring that $\mu_1 < \cdots < \mu_k$. The vector of weights $w = (w_1, \ldots, w_k)$ is given a Dirichlet prior, $w \sim D(\delta, \ldots, \delta)$. Again following Richardson & Green (1997) a $\Gamma(g, h)$ prior is placed on the hyperparameter β and k is given a Uniform prior over the values $1, \ldots, k_{\max}$, for $k_{\max} = 30$. The fixed parameters take the following values $\xi = 21.73$, $\kappa = 0.0016$, $\alpha = 2$, $g = 0.2$, $h = 0.016$ and $\delta = 1$.

Constructing the Chain

Richardson & Green (1997) use 7 types of moves to update the Markov chain. Five of the moves keep the dimension fixed and update the parameters μ, σ, w, β and z using the univariate full conditional distributions. These full conditional distributions are given in detail in Richardson & Green (1997). Due to the ordering constraint on μ each μ_j is updated using a Metropolis-Hastings step where the proposal is its full conditional distribution, which is Normal, and the proposal is rejected if it does not satisfy the ordering constraint. The parameter β and each σ_j^{-2} have Gamma distributions as their full conditionals and the weight vector w has a Dirichlet full conditional distribution. Each z_i can take one of the values $1, \ldots, k$, and the probability of each possible value can be directly calculated. So each z_i is easily updated according to its conditional distribution.

For the dimension changing moves Richardson & Green (1997) constructed two different types of move, a birth/death move and a split/merge move, which are now briefly described. There are two parts to each move because for a particular move the reverse move must be constructed and taken into account in the acceptance probability, (4.2).

A birth consists of proposing a new component l with parameters (μ_l, σ_l^2) drawn from the prior and a weight w_l drawn from a Beta distribution. For the reverse move which is a death, an empty component is randomly chosen to be removed. The construction of the moves and the acceptance probabilities are both kept simple by not

reallocating any of the observations under these moves. In both moves the weights are rescaled to sum to 1. The acceptance probabilities for both a birth and a death are given in Richardson & Green (1997).

The merge move consists of choosing two components with adjacent means and proposing to replace them with one component whose mean and variance are functions of the means and variances of the components to be merged. When matching dimensions in generating the reverse split move, 3 random variables are used because the extra component created has three more parameters. The means, variances and weights of the new components are functions of the old components parameters and the 3 generated random variables. The weights in both moves are rescaled to sum to 1. For the split move any observations allocated to the component being split are reallocated among the two new components. The acceptance probability for the split/merge move is given in Richardson & Green (1997).

Richardson & Green (1997) construct a sampler from these constituent parts by updating w, μ, σ^{-2}, z and β at each iteration using their conditional distributions. Also at each iteration they randomly choose to make a split or merge move with probabilities b_k and $d_k = 1 - b_k$ respectively where $b_{k_{\max}} = 0$, $b_1 = 1$ and $b_k = 0.5$ for $k = 2, \ldots, k_{\max}-1$. Similarly at each iteration a birth or death is proposed with probabilities b_k and d_k respectively.

Coupling Chains

In all the examples seen so far the procedure for coupling chains so that they use the same sequence of random variates at each iteration has been straightforward because in order to update a chain from one iteration to the next only a finite number of random variates at each iteration has been required, whether they are Uniform, Normal or Gamma variates. This is no longer the case when simulating from the model described by Richardson & Green (1997), and indeed we do not want to be restricted to examples which require a finite input of random variates at each iteration. To make it clear how we went about coupling chains, we now describe in detail the construction of the coupling, in particular we make it clear how the chains were coupled using sequences of $U(0,1)$ variates even though the dimension of the state of the chains vary. In the particular example from Richardson & Green (1997) the number of components is restricted to a maximum of 30, however in general the number of components may not be restricted, so in the following discussion we will consider the situation where the number of components is unbounded.

The states of several seeds for $U(0,1)$ random number generators are updated at each iteration, and these are used to update the state of the coupled chains. We now

look at how these seeds are used.

We first look at the fixed dimension moves, which update chains using the conditional distributions. The set of parameters which make up the state of a chain splits into convenient subsets μ, σ, w, β and z which are dealt with separately. To update the univariate components in each of these subsets we keep track of a seed which changes at each iteration. At a particular iteration this one seed is used to generate a sequence of $U(0,1)$ variates which in turn is used to update the subset of parameters. All the methods used to generate samples from Normal, Gamma, Beta and Dirichlet distributions were taken from Devroye (1986).

At each iteration the state of a chain will have $\mu = (\mu_1, \ldots, \mu_k)$ as a subset of its current set of parameters. Each μ_j is updated using a Metropolis-Hastings step by first obtaining a sample from its conditional distribution which is,

$$\mu_j|\cdot \sim N\left(\frac{\sigma_j^{-2}\sum_{i:z_i=j} y_i + \kappa\xi}{\sigma_j^{-2}n_j + \kappa}, (\sigma_j^{-2}n_j + \kappa)^{-1}\right)$$

and then accepting this proposal if the ordering restriction, $\mu_1 < \mu_2 < \cdots < \mu_k$, holds. In the scheme we implemented, to update each μ_j requires two $U(0,1)$ variates, which are used to generate a $N(0,1)$ variate and then this is transformed in order to get a sample from the conditional distribution of μ_j. At iteration n we have the seed for a $U(0,1)$ random number generator, $s_\mu^{(n)}$. At time n $s_\mu^{(n)}$ is used to generate a sequence of Uniform variates ${}^1u_\mu^{(n)}, {}^2u_\mu^{(n)}, \ldots$ which is used by every chain. This construction is illustrated in Figure 4.12. To update μ_j we simply use ${}^{2j-1}u_\mu^{(n)}$ and ${}^{2j}u_\mu^{(n)}$. The potentially infinite sequence of ${}^lu_\mu^{(n)}$ ensures that there are enough random variates available regardless of the value of k.

The state of a chain also contains the subset of parameters $\sigma^{-2} = (\sigma_1^{-2}, \ldots, \sigma_k^{-2})$, and the conditional distribution of each of these is

$$\sigma_j^{-2}|\cdot \sim \Gamma\Big(\alpha + \frac{n_j}{2}, \beta + \sum_{i:z_i=j}(y_i - \mu_j)^2\Big) \qquad j = 1, \ldots, k. \tag{4.4}$$

The method we use to generate a sample from these distributions at time n is, for each j first generate $v_{\sigma^{-2}}^j \sim \Gamma(\alpha + n_j/2, 1)$ and then return $v_{\sigma^{-2}}^j/(\beta + \sum_{i:z_i=j}(y_i - \mu_j)^2)$ which is a sample from (4.4). We used a rejection algorithm to generate each $v_{\sigma^{-2}}^j$ which needs an infinite sequence of Uniform variates. To couple chains so that each chain uses the same Uniform variates in the same order, a seed $s_{\sigma^{-2}}^{(n)}$ was used at each iteration. The seed $s_{\sigma^{-2}}^{(n)}$ was used to generate k seeds, $s_{\sigma^{-2}}^{(n,j)}$ for $j = 1, 2, \ldots$. The j^{th} seed is used to generate a sequence of Uniform variates ${}^iu_{\sigma^{-2}}^{(n,j)}$, $i = 1, 2, \ldots$. This sequence of Uniform variates is then taken as the input to the rejection algorithm used to generate $v_{\sigma^{-2}}^j$.

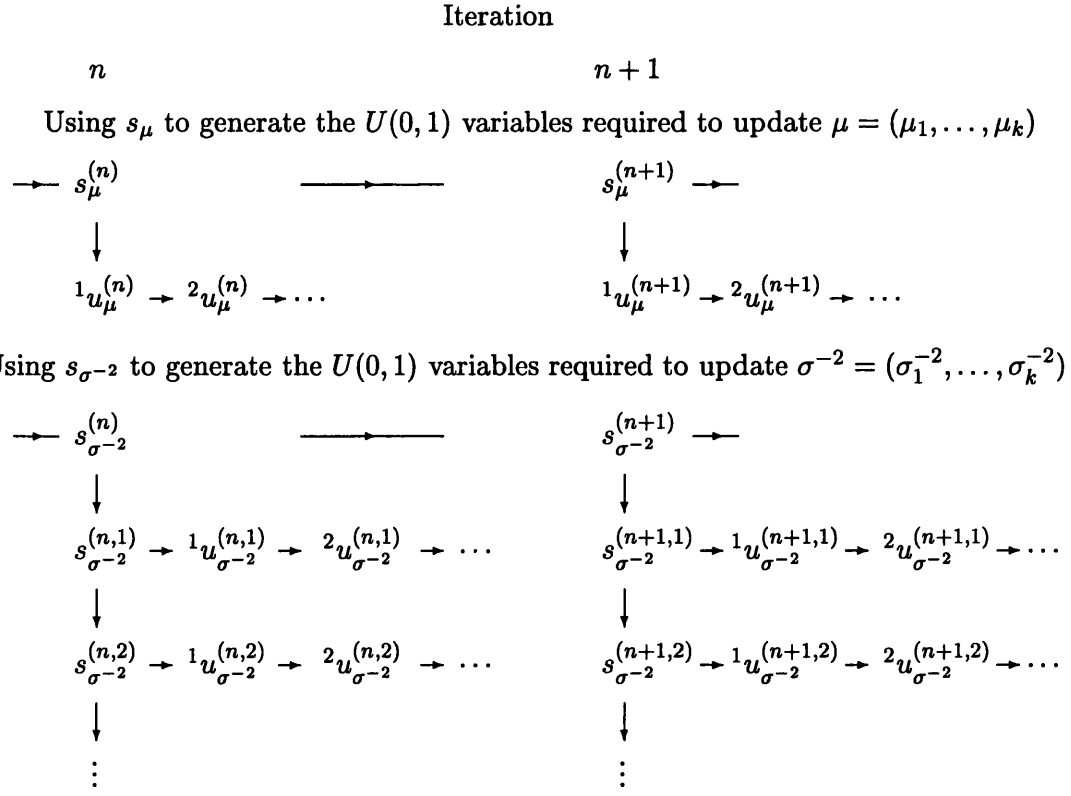

Figure 4.12: Diagram illustrating the use of seeds at each iteration to update the parameter vectors μ and σ^{-2}. Each ${}^iu_\mu^{(n)}$ denotes the i^{th} Uniform variate generated using $s_\mu^{(n)}$. The $s_{\sigma^{-2}}^{(n,i)}$ refer to the i^{th} seed generated at iteration n from $s_{\sigma^{-2}}^{(n)}$, and ${}^ju_{\sigma^{-2}}^{(n,i)}$ is defined as the j^{th} Uniform variate generated using this seed.

This method for producing the required sequences of Uniform variates is demonstrated diagrammatically in Figure 4.12.

One thing to consider under this scheme is that sometimes two seeds are generated from a single seed, so here we look at the example of the seed $s_{\sigma^{-2}}^{(n)}$ which is used to generate both $s_{\sigma^{-2}}^{(n+1)}$ and $s_{\sigma^{-2}}^{(n,1)}$. When programming the pseudo random number generators which generate the sequences of seeds we need to make sure that $s_{\sigma^{-2}}^{(n+1)}$ and $s_{\sigma^{-2}}^{(n,1)}$ have as little dependence as possible. To generate the seeds for σ^{-2} shown in Figure 4.12 three different pseudo random number generators were used. The first (generator A) generated the seeds $s_{\sigma^{-2}}^{(n)}$, the second (generator B) generated the seeds $s_{\sigma^{-2}}^{(n,j)}$ and the third (C) generated the Uniforms ${}^iu_{\sigma^{-2}}^{(n,j)}$. At iteration n generator A was used twice to generate two seeds, (a function of each of these seeds leading to a Uniform); the Uniform produced from the first seed was used to uniformly generate a starting seed for generator B, then $s_{\sigma^{-2}}^{(n+1)}$ was set equal to the second seed. Starting seeds for generator C were generated from B in the same way.

The vector of weights $w = (w_1, \ldots, w_k)$ is updated using its Dirichlet conditional distribution,

$$w|\cdot \sim D(\delta + n_1, \ldots, \delta + n_k). \tag{4.5}$$

To generate a sample from this distribution we generated the vector of random variates $(u_w^1, \ldots, u_w^k)$, where each $u_w^j \sim \Gamma(\delta + n_j, 1)$ was generated using our standard method for generating samples from a Gamma distribution using a rejection algorithm. A sample from (4.5) is obtained by setting each

$$w_j = \frac{u_w^j}{\sum_{l=1}^k u_w^l} \qquad j = 1, \ldots, k.$$

At each iteration we keep track of the seed s_w, which is used to generate k seeds, and each of these in turn generates the sequence of Uniform variates required by the rejection algorithm used to generate each u_w^j.

The parameter β has conditional distribution,

$$\beta|\cdot \sim \Gamma\Big(g + k\alpha, h + \sum_{j=1}^k \sigma_j^{-2}\Big).$$

Our standard rejection sampler for the Gamma density is used to simulate from this distribution by using the sequence of Uniform variates generated using the seed s_β.

The vector of allocation parameters $z = (z_1, \ldots, z_n)$ is easily updated using n Uniform variates because

$$\mathbb{P}(z_i = j|\cdot) \propto \frac{w_j}{\sigma_j} \exp\left(-\frac{(y_i - \mu_j)^2}{2\sigma_j^2}\right) \qquad i = 1 \ldots, n,$$

and so each z_i can be updated using its inverse cumulative distribution function. At each iteration the seed s_z is used to generate the n Uniforms required.

The moves which change dimension require no extra methods outside of those used by the fixed dimension moves. Each pair of dimension changing moves is simply coupled by generating random variates based on a single seed at each iteration.

We use the seed s_B to generate the sequence of Uniform variates needed to implement the birth/death move. The first Uniform will be used in the acceptance step. The second Uniform in the sequence is used to decide whether to propose a birth or a death. The third Uniform is used by the death to choose an empty component. Generating a birth requires one Normal, one Gamma and one Beta random variate. So the next two Uniforms are used to generate the Normal, the Gamma variate is generated independently of the state and so the number of Uniforms required is the same for

every chain, and then finally the remaining Uniforms in the sequence are used to generate the Beta$(1, k)$ variate which involves using a rejection algorithm and so requires an unbounded number of Uniforms, the number of which depends on the state of the chain.

The split/merge move requires the generation of one Uniform variate used in the acceptance step, two Beta$(2, 2)$ variates, and a Beta$(1, 1)$ variate. These are all generated independently of the state and so can be generated in order using the sequence of Uniform variates obtained using the seed s_S.

To couple chains ACFTP keeps track of the state of the seeds s_μ, $s_{\sigma^{-2}}$, s_w, s_β, s_z, s_B, s_S at each iteration. This ensures that once two chains have coalesced, they will remain together at all iterations thereafter. It also helps chains get closer together because both chains will use some of the same variates in their updates even if they are in different states.

Despite sounding rather elaborate it does not require a lot of extra programming. Taking a structured approach to writing the programs means that little extra problem specific programming is required above that needed to run a chain forward in time. Most of the mechanisms described will be in place simply for running chains forward. Certainly any perfect simulation algorithms will need such housekeeping to keep track of seeds in order to couple chains. The programs used to generate the results shown in this thesis separated the common code used to implement the ACFTP algorithm from the code specific to a particular problem. The ACFTP code keeps track of seeds, generates samples from standard distributions using seeds, and runs chains between the desired iterations. This code only needs to be written once. The code for each specific model, as well as containing the usual state variable describing the components of a chain, will also have to contain a state variable for the seeds and a function for generating new seeds at an iteration.

Coalescing Chains

The z_i are discrete variables so these are simply left to coalesce on their own. The conditional distribution of w given in (4.5) depends only on the discrete values of $n_1, \ldots, n_k$ so w can also be left to coalesce on its own. The univariate components of the parameters β, μ and σ^{-2} will be coalesced using Neal's (2002) method and for this we need to choose a vector ω containing the widths of the intervals used in the Uniform proposal.

The fact that the length of the vector μ is not fixed means that the choice of the interval width in the Uniform proposals has to reflect this. Specifying a different interval width for every μ_j for $j = 1, \ldots, k_{\max}$ will not be of benefit. Each μ_j can only

be identified by its labelling according to the ordering of the μ_js, and after a move that adds or removes a component this ordering will change, so the distribution of the simulated values of a particular μ_j will change depending on the ordering. This is the well known problem of identifiability in distributions where the dimension of the parameter vector changes, (see for example Richardson & Green (1997) and Hurn et al. (2003)). For this reason we use one interval width ω_μ for every component of μ. Similarly for each σ_j^{-2} we use the same value of $\omega_{\sigma^{-2}}$. We have already seen how robust the coalescence times are to the choice of ω for previous examples in Sections 4.4.1 and 4.4.2 so we believe that this choice should not greatly effect the speed with which chains coalesce. For the parameter β we use ω_β.

A search for values of $\omega = (\omega_\mu, \omega_{\sigma^2}, \omega_\beta)$ that lead to an overall acceptance rate between 0.2 and 0.6 suggested values of around 0.1 for each of ω_μ, ω_{σ^2} and ω_β, when using $L = 20$ and $\mathcal{L}_2$. Figure 4.13 shows how this choice compares to other choices. Each graph plots the mean number of iterations taken for two chains to coalesce against a component of ω on the $\log_{10}$ scale. The means were obtained from 1000 runs. The values in each graph give the average coalescence time across the ω component missing from that graph. From Figure 4.13 it is clear that choosing $\omega = (0.1, 0.1, 0.1)$ leads to reasonably quick coalescence. Even though the choice of ω_μ matters little and we could use any sensible value, choosing values of $\omega_{\sigma^2} = 0.1$ and $\omega_\beta = 0.1$ would be around optimal, as can be seen by the fact that larger values do very much worse and smaller values also take a little longer to coalesce chains. These graphs are on the log scale so the coalescence time is still robust to the choice of ω and even values of ω such as 100 would not be very poor. To obtain the results shown in the rest of this section we alternated between 20 values of ω of the form $(\omega_i, \omega_i, \omega_i)$, where $\omega_i = 0.01 + i(1 - 0.01)/19$, $i = 0, \ldots, 19$, which simply means alternating between 20 equally spaced values between 0.01 and 1.

The choice of L and $\mathcal{L}$ has also been looked at. We follow the recommendations from Section 4.3.2 which suggest choosing a value of L anywhere in the range 30 to 100, so the results in the rest of this section were obtained using $L = 30$, and $\mathcal{L} = \{0, 1\}$. Simulation results suggest that these are sensible choices, and that the relationship between coalescence times and choice of L and $\mathcal{L}$ in this mixture distribution is the same as that observed in earlier examples.

The Performance of ACFTP

We still require a starting distribution to be chosen. Here we choose to use the prior as the starting distribution. This is because the prior should be chosen so that it covers all the areas that we would expect the posterior to give significant probability

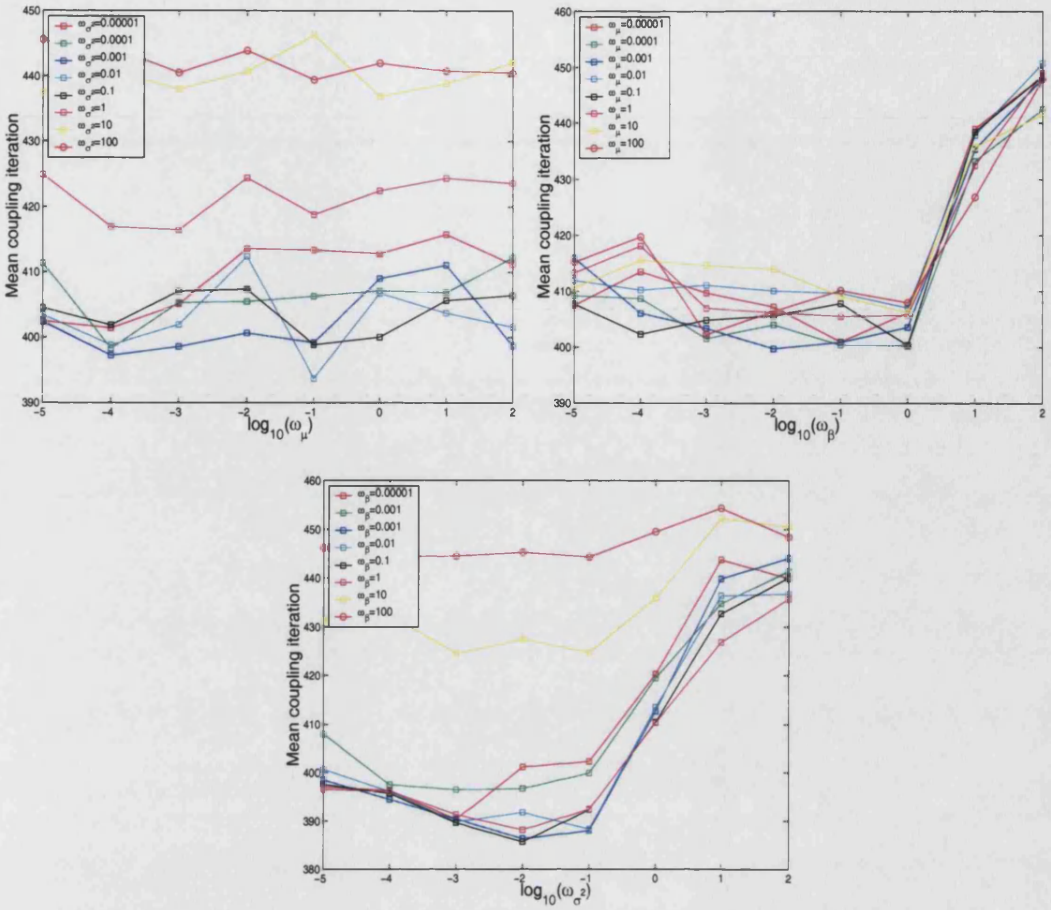

Figure 4.13: Mean coalescence time for two chains for different combinations of ω_μ, ω_{σ^2} and ω_β (on the log scale) for the univariate mixture of Normals. In each graph the times were averaged across the component of ω not appearing.

to, if the posterior placed significant probability on areas not done so by the prior then we should be suspicious about the choice of prior. We could choose starting values in other ways. For example Gelman & Rubin (1992) suggest constructing an over-dispersed approximation to the desired distribution using a mixture of multivariate t-distributions, as mentioned in Section 1.5. This approach is complicated by the varying dimension which would mean that for every k a separate over-dispersed distribution is needed. Certainly when looking at a new problem some preliminary work is always sensible before running a chain to simulate from a distribution. We will see however that the choice of the prior as the starting distribution is fine here. It is also the case that the birth move generates the proposal for the mean and variance of a new component from the prior and it has a good acceptance rate, which suggests that the

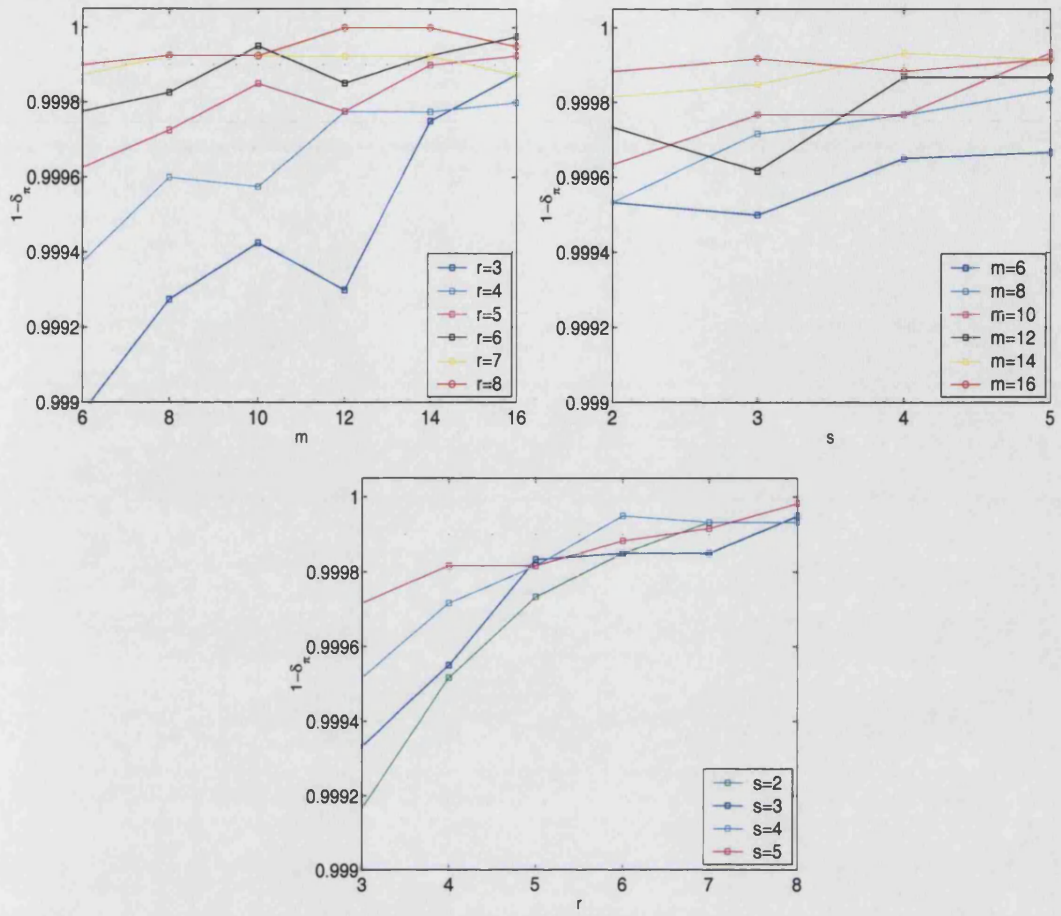

Figure 4.14: Proportion of stationary chains coalescing with ACFTP chains from 200 runs for different combinations of m, r and s for the mixture of Normals.

prior does provide an over-dispersed estimate of the posterior.

The correctness of the ACFTP procedure was confirmed by running forward samples starting in the states returned by ACFTP, and checking that the estimates of the distributions of parameters agreed with those given in Richardson & Green (1997).

Figure 4.14 shows how well ACFTP did at returning suitable states at time 0. Each graph plots the proportion, $1 - \delta_\pi$, of coupled chains starting in a state sampled from the stationary distribution coalescing with the ACFTP chains against each value of m, r and s. This proportion is averaged over the parameter not appearing in the graph. To serve as samples from the stationary distribution one chain was run forward in time to collect 16000 samples spaced 10000 iterations apart. The long length of these runs and separation of the samples should mean that they can be regarded as samples from the stationary distribution. For any combination of m, r and s in the range shown

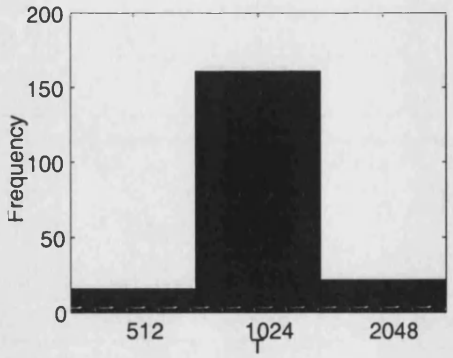

Figure 4.15: Histogram of the number of iterations in the past that ACFTP went back to from 200 runs using $m = 10$, $r = 5$ and $s = 3$, for the mixture of Normal distributions.

this proportion is greater than 0.999. The graphs show that the recommendations for choosing values of m, r and s obtained by looking at multivariate Normal examples perform as well as predicted.

A histogram of the number of iterations that ACFTP went back to in the past is given in Figure 4.15 suggesting that most chains will forget their initial states after around 2000 iterations. The computing time taken by ACFTP obviously varies depending on how far in the past it needs to go, but runs that go 1024 iterations in the past took around 70 seconds[3] with $m = 12$, $r = 5$ and $s = 3$.

Sensitivity to the Starting Distribution

We have also looked at how well ACFTP does when the starting distribution is poorly chosen and as in previous examples we consider using a point mass starting distribution to demonstrate a worse case scenario. The state for this distribution was taken as one of the states from the very long runs used as samples from the stationary distribution. ACFTP was repeated for various values of r and s and the results are shown in Figure 4.16. These results compare well with those from previous examples when using a point mass starting distribution, (for example Figure 4.6). Using a suitable value of r, such as 4 or 5, and s, such as 3, the proportion of stationary chains that coalesce to the returned sample at time 0 is around 0.998.

This should represent the worst possible outcome. When all the initial m chains start from the same state these chains immediately coalesce. So in Step 2 of ACFTP (Section 3.2) these chains will have been run from time -2, and r samples at time 0

[3]Using a 400MhZ UltraSPARC II processor.

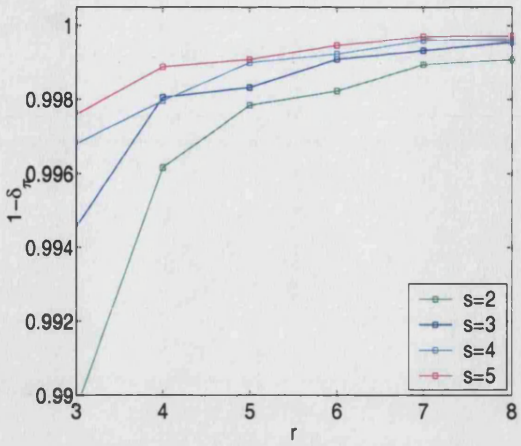

Figure 4.16: Proportion of stationary chains coalescing with ACFTP chains for different combinations of r and s when using a point mass starting distribution, for the mixture of Normals.

obtained. Returning a value of $T = -2$ in Step 2 means that the samples from the forward chains are taken at times $0, 2, \ldots, 2(s-1)$, and so for values of s around 3 the forward chains will only be run for a few iterations. Across all r repetitions the forward chains will be simulating from the same area of the state space and so they would be expected to coalesce quickly with the initial chains, resulting in ACFTP finishing after one cycle, and returning r states from close to the point mass starting distribution as supposed samples from the stationary distribution. The results from Figure 4.16 show that this is not the case, and that during the ACFTP procedure the chains spread out throughout the state space covering a high proportion of it.

Until now we have forgone a discussion of the choice of the starting distribution and how it can lead to a failure to correctly diagnose convergence. In using a point mass starting distribution as our worse case scenario, and in general when looking at the performance of ACFTP, we are assuming that the usual problems with choosing starting states for chains have been overcome. When using any diagnostic it is hoped that before finally running a chain any problems that the chain could have in exploring the state space have been solved, for example freely moving between modes. We also assume that the starting distribution has been chosen to include any appropriate areas of the stationary distribution that might be missed in the diagnostic process because of the chain not moving easily throughout the state space. This could be isolated modes or tails of the distribution that are not easily reached. This is common to all diagnostic methods because convergence can be prematurely diagnosed if the chains do not move around the whole state space.

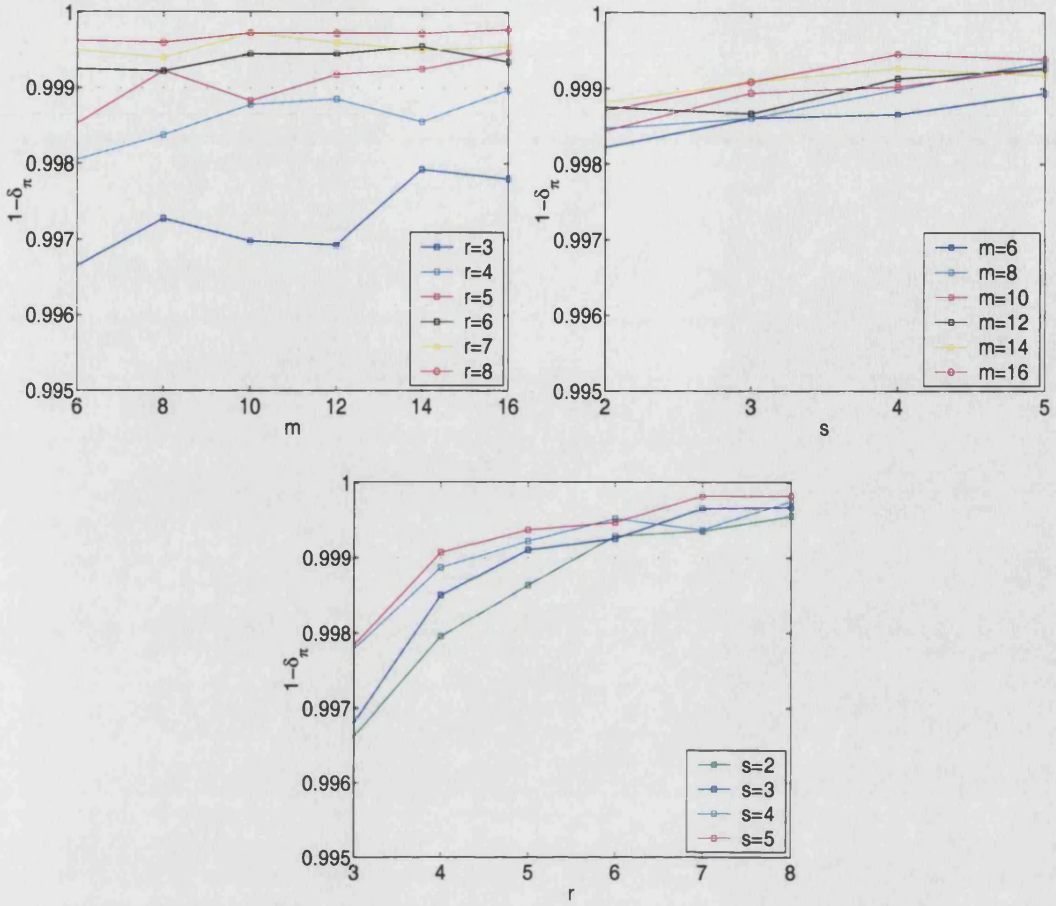

Figure 4.17: Proportion of stationary chains coalescing with ACFTP chains in the mixture of Normals for different combinations of m, r and s when the starting distribution is taken as the stationary distribution.

If we assume that a starting distribution has been found that covers the areas given weight by the stationary distribution, then we can further investigate the sensitivity of ACFTP to the starting distribution by considering a starting distribution that is not over-dispersed, and yet not as bad a choice as a distribution with all its mass at one point. We look at using the stationary distribution as our starting distribution, as we already have an idea about how ACFTP performs with a very poor choice of starting distribution from Figure 4.16. To implement this in practice the samples from the starting distribution were taken from very long runs of a chain, in the same way that samples from the stationary distribution are obtained when testing the performance of ACFTP.

The results from running ACFTP with the starting distribution equal to the sta-

tionary distribution are shown in Figure 4.17. If the recommended values of m, r and s are chosen then the proportion of chains started from the stationary distribution will be close to 0.999. Despite not performing as well as when using an over-dispersed starting distribution, (as in Figure 4.14), ACFTP is only failing to coalesce chains from the stationary distribution one in a thousand times. From a perfect simulation point of view, we could interpret this as, ACFTP will coalesce chains starting from a subset of the state space that contains 99.9% of the stationary distribution. Given that this figure was obtained without using an over-dispersed starting distribution we would hope that with a better starting distribution we can do at least as well.

4.7.7 A Mixture of Regressions

Description of Model

In this section we apply ACFTP to a regression model looked at by Hurn et al. (2003). We follow exactly the approach taken by Hurn et al. (2003) when they fit this model to the CO_2 dataset which consists of the gross national product (GNP) per capita in 1996 x, and the estimated CO_2 emissions per capita in the same year y. Data from 27 countries are looked at. The construction of the model is similar to that given in Section 4.7.5. Each observation $(x_1, y_1), \ldots, (x_n, y_n)$ consists of a quantity y_i depending on a covariate x_i. The structure of the dependency is given by the weighted mixture of distributions,

$$y_i|x_i \sim \sum_{j=1}^{k} w_j f_j(y_i|\theta_{0j}, \theta_{1j}, \tau_j, x_i),$$

where,

$$f_j(y_i|\theta_{0j}, \theta_{1j}, \tau_j, x_i) = \frac{\tau_j}{\sqrt{2\pi}} \exp\left\{-\frac{\tau_j}{2}(y_i - \theta_{0j} - \theta_{1j}x_i)^2\right\}.$$

The parameter vector $\theta_j = (\theta_{0j}, \theta_{1j}, \tau_j)$ consists of the intercept, slope and precision of the j^{th} regression line. Independent Normal priors are put on θ_{0j} and θ_{1j}, Exp(1) priors on the τ_j and a Dirichlet prior on the weight vector w. We use the same Markov chain construction used by Hurn et al. (2003) which is now described.

Constructing the Chain

Hurn et al. (2003) use Stephens's (2000) birth/death procedure for changing the number of components in the mixture and implement it as described in Section 4.7.3. They run a standard Gibbs sampler (with a fixed k) updating each of θ_{0j}, θ_{1j}, τ_j and z according to their conditional distributions for $I = 10$ iterations and then the birth/death process is run with $t_0 = 1$. One cycle of this counts as one iteration of their sampler.

Coupling and Coalescing Chains

Chains are easily coupled and can be coalesced using Uniform proposals. For the Gibbs sampler updates with a fixed number of components the conditionals distributions are available and chains can be coupled using the method described in Section 4.7.6. To coalesce chains exactly Neal's (2002) Uniform proposals are used on each θ_{0j}, θ_{1j} and τ_j. The weights are left to coalesce on their own because their conditional distribution depends only on discrete parameters.

Coupling chains during the birth/death process is straightforward and follows the same method of using sequences of $U(0,1)$ variables described in Section 4.7.6. For the particular example here only one sequence of Uniforms is required for the birth/death process because the number of Uniform variables required at each step of the process is independent of the current state of the chain. In the algorithm given in Section 4.7.3, one $U(0,1)$ is required in step 3 to get the time to the next birth or death, one in step 4 to choose between a birth or a death, and then five in step 5 which are either all used to generate a sample from the prior for a birth proposal or one is used to choose a component to remove in the case of a death. This coupling method means that two coupled chains choosing a birth at the same point in the sequence of births and deaths will propose a birth to the same state which aids the coalescence of chains.

For the same reasons as given in Section 4.7.6, when discussing the mixture of univariate Normal distributions, we take $\omega = (\omega_{\theta_0}, \omega_{\theta_1}, \omega_\tau)$, so that every component of the same type uses Uniform proposals with the same width. A search for values of ω yielding a product of acceptance probabilities between 0.2 and 0.6 shows that suitable values for any component of ω lie around 0.1. Different choices were looked at and any combination of components of ω taking values in the range 0.0001 to 1 was found to coalesce two chains in around 200 iterations. So for the results in this section we alternate between 20 values of ω between $(0.01, 0.01, 0.01)$ and $(1, 1, 1)$.

For convenience we simply do one Uniform proposal in every I iterations between the birth/death moves. When approaching a new problem we would suggest following the recommended method of choosing L and $\mathcal{L}$ as previously discussed. Some of the coalescing is done through the birth/death process and we could consider not using any Uniform proposals at all, so unlike the fixed dimension examples no special methods are required to get chains to coalesce. The effect of the Uniform proposals is easily seen by starting two chains from states sampled from the prior and looking at the average number of iterations taken to coalesce. From 1000 runs it took an average of 171 iterations for two chains to coalesce without using any special methods, but only 41 iterations using Uniform proposals, which demonstrates that it is worthwhile using special steps to coalesce chains. The birth/death steps are also much more computationally demanding

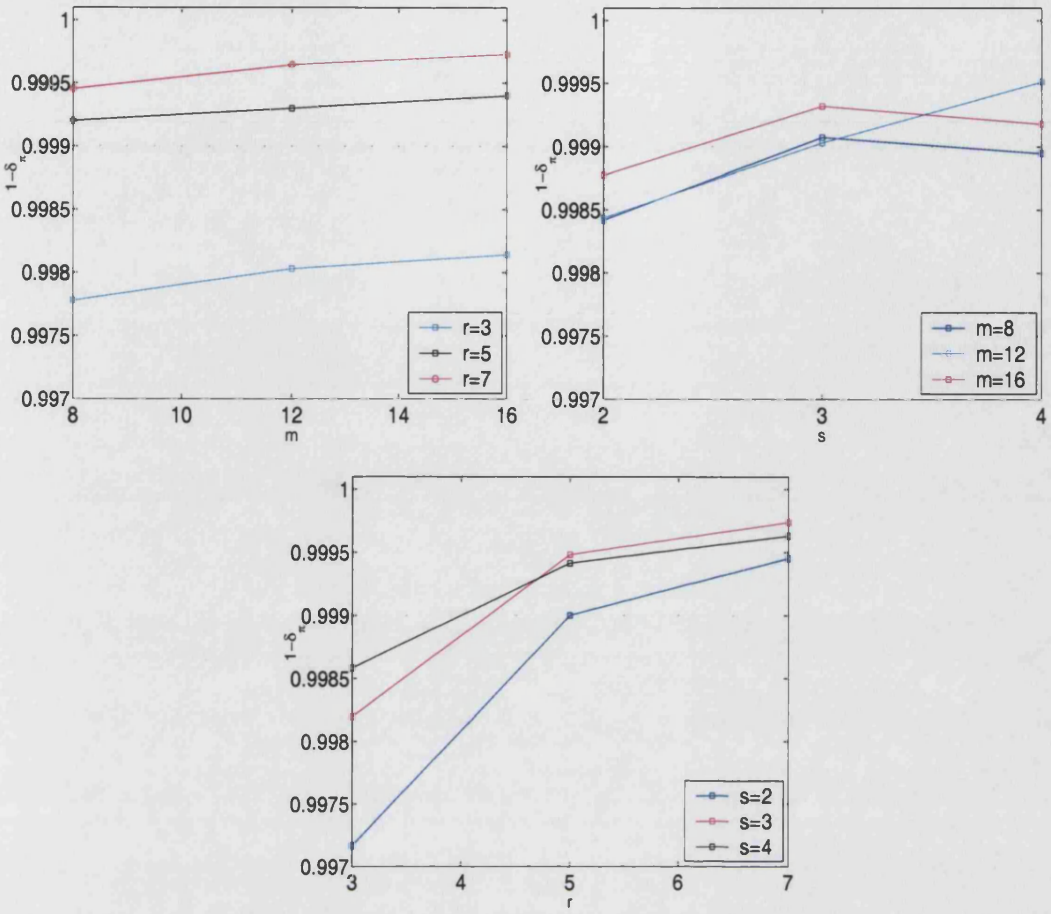

Figure 4.18: Proportion of stationary chains coalescing with the ACFTP chains for different combinations of m, r and s for the mixture of regressions.

and so one would not want to use them more than necessary.

Applying the ACFTP Procedure

As in the previous examples ACFTP was used with different values of m, r and s, and we looked at the proportion of times that chains started from states obtained from very long runs coalesced to the same state at time 0 as returned by ACFTP. The same approach to the starting distribution was taken as before, the prior was used, although as mentioned in the other examples a more careful approach to choosing starting states should be taken when looking at a new problem. The proportion of stationary chains coalescing with the ACFTP chains is shown in Figure 4.18, where the proportions in each graph are obtained by averaging over the results for the parameter not included.

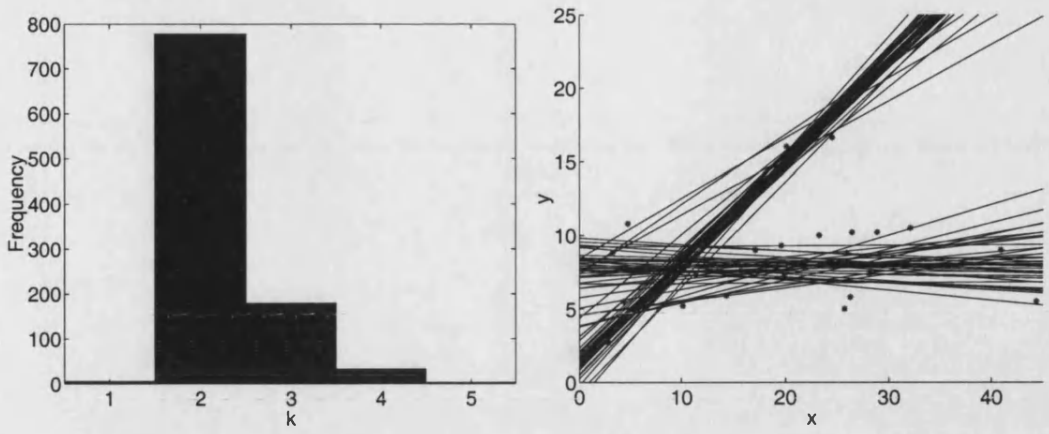

Figure 4.19: *(left)* Histogram of simulated values of k. *(right)* Fitted lines from simulations where $k = 2$.

Even though these proportions are not as high as in previous examples, stationary chains are only failing to coalesce with the ACFTP chains on about 1 in 1000 attempts. For the particular choice of $m = 12$, $r = 5$ and $s = 3$ which would be a recommended selection based on previous examples, the proportion was 0.99945.

Figure 4.19 shows some results obtained from this model, which agree with the simulation results given in Hurn et al. (2003). These plots were obtained by running ACFTP with several repetitions and then obtaining samples from forward chains starting in the states returned by ACFTP. The histogram of sampled values of k shows that the model places most probability on $k = 2$. For many of the samples where $k = 3$ one of the regression lines was given little weight and only had a few, if any observations allocated to it so these samples could also be regarded has having two regression lines. Fitted lines from 500 samples are also shown plotted with the data.

Sensitivity to the Starting Distribution

We have also looked at the sensitivity of ACFTP to the choice of starting distribution. We first consider our worst case situation where the starting distribution consists of a point mass. A sample from one of the long runs which looked to be close to a mode was used, and so all chains in the starting distribution started from this one state. The proportion of stationary chains coalescing with the ACFTP chains for various combinations of r and s is shown in Figure 4.20. These results are similar to those from the mixture of Normals example using a point mass starting distribution (Figure 4.16), with only around 1 in every 250 stationary chains not coalescing with the ACFTP chains.

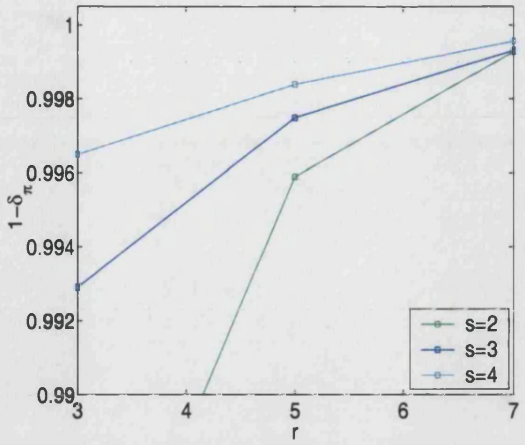

Figure 4.20: Proportion of stationary chains coalescing with ACFTP chains when using a point mass starting distribution for different combinations of r and s for the mixture of regressions.

In looking at a distribution which is not over-dispersed, but not as bad a choice as starting all the initial chains in the same state, we take the stationary distribution itself as the starting distribution. The results from doing this are shown in Figure 4.21. Again these results are similar to those of the mixture of Normals (Figure 4.17). Suitable choices of m, r and s lead to around 1/1000 stationary chains failing to coalesce with the ACFTP chains. This is only slightly worse than the results obtained using the prior as the over-dispersed distribution, and we can conclude that any reasonable choice for the starting distribution should perform satisfactorily.

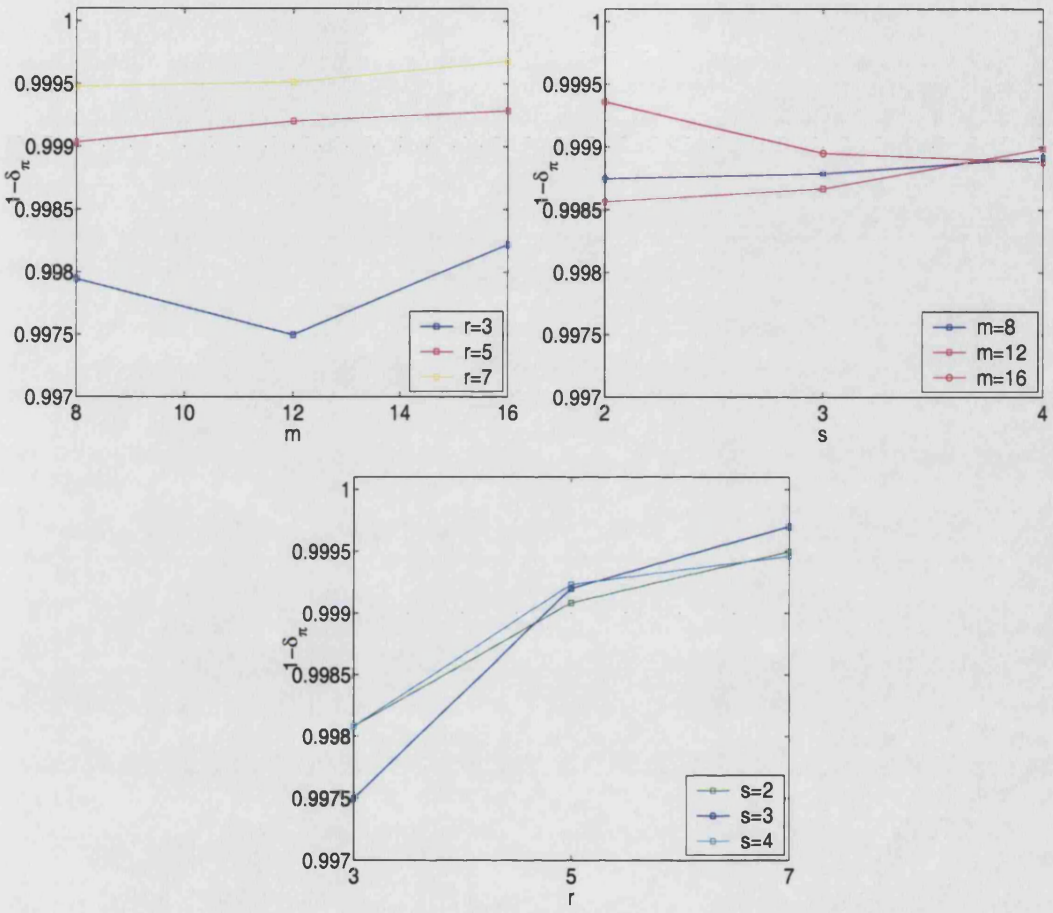

Figure 4.21: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s when the starting distribution is taken as the stationary distribution.

4.8 A High Dimensional Discrete Problem

4.8.1 Introduction

Propp & Wilson (1996) originally looked at applying their coupling from the past algorithm to distributions arising in statistical physics which have certain properties. Their algorithm required that the distribution of interest exhibit a partial ordering with a minimum and maximum state, and that an SRS can be constructed which updates a chain simulating from the distribution and which preserves the partial ordering at each iteration. Propp & Wilson's (1996) algorithm has been extended to cope with a much wider range of distributions, but some distributions still remain a problem.

One such example is the antiferromagnetic Potts model, which is a distribution defined on an array of cells, where each cell can take one of $C \geq 3$ possible colours. The Potts model is an extension of the Ising model to more than two colours which was one of the first applications of Propp & Wilson's (1996) coupling from the past. In the antiferromagnetic Potts model neighbouring cells tend to be of a different colour. Huber (1998) suggests a way to generate perfect samples from, among other distributions, the antiferromagnetic Potts model using summary states. This method is described in Section 4.8.3. When generating perfect samples from the antiferromagnetic Potts model using this method, Childs, Patterson & MacKay (2001) report that the method slows down greatly at a temperature above the temperature where the Gibbs sampler suffers from a critical slowing down when simulating from this model. In this section we look at applying ACFTP to the antiferromagnetic Potts model at temperatures for which it is impractical to use Huber's (1998) summary states method. This will serve as an example of applying ACFTP to a different kind of distribution to that seen in previous sections, and demonstrate that ACFTP is a viable method for generating samples from complex finite distributions.

4.8.2 The Potts Model

In this section we look at the Potts model defined on an $N \times N$ square lattice, $X = \{X_{ij}; i = 1, \ldots, N, j = 1, \ldots, N\}$, where each cell in the lattice can take a value in $\mathcal{C} = \{1, \ldots, C\}$ for $C \geq 3$. Each value in $\mathcal{C}$ is often referred to as a colour, because it is a generalization of the Ising model which takes $C = 2$ and is sometimes considered as a distribution on an array of black and white cells. The Potts model is defined by the distribution π on an N dimensional lattice where

$$\pi(x) = \frac{1}{Z} \exp\Big\{ -\frac{1}{2}\beta \sum_{<ij,kl>} I[x_{ij} \neq x_{kl}] \Big\}$$

and the sum is over all neighbouring pairs of cells. In order to compare our results with those of Childs et al. (2001) we have included the factor of $\frac{1}{2}$ and take neighbouring pairs to mean cells that lie next to each other in either a horizontal or vertical direction. The large number of possible configurations means that calculating the normalizing constant Z will be impractical, and so to investigate the behaviour of this model we need to be able to simulate from π. The conditional distribution of the value in each cell is available and is a function of only C, β and the states of the neighbouring cells. Therefore the Potts model can be conveniently simulated from using the Gibbs sampler, updating one cell at a time. The conditional distribution of cell (i,j) given all other cells is simply

$$\pi(x_{ij}|x_{-(ij)}) \propto \exp\Big\{-\frac{1}{2}\beta \sum_{(k,l)\in\delta(i,j)} I[x_{ij} \neq x_{kl}]\Big\}, \tag{4.6}$$

where $x_{-(ij)}$ denotes the lattice less the (i,j)th cell and $\delta(i,j)$ is the set of cells in the neighbourhood of cell (i,j). When $\beta > 0$ this distribution is attractive (ferromagnetic) so that states with similar neighbouring cells are given higher probabilities, whereas when $\beta < 0$, the non-attractive case (antiferromagnetic), states with neighbours of differing colours are preferred. The value of $|\beta^{-1}|$ is referred to as the temperature of the system. At low temperatures the Gibbs sampler will suffer from poor mixing because of the strong correlations between cells.

4.8.3 The Summary States Method

Propp & Wilson's (1996) original CFTP algorithm involves following two chains which are started at the minimum and maximum states, and being able to construct these chains to be monotonic so that all other possible chains are sandwiched between them. For many distributions a monotone chain cannot be found. Huber (1998) describes a method for obtaining a perfect sample from the anti-ferromagnetic Potts model using a single chain whose state summarizes one's knowledge of the system. This single chain is run on a modified state space from sufficiently far in the past that the state of this chain at time 0 will be the same as that of all chains started from the same time in the past.

The summary states method is implemented by introducing a new possible state for each component, and running a modified Markov chain which has this additional state. In addition to the possible states $1,\ldots,C$ an extra state ? is introduced. From a CFTP perspective, a component of the summary chain being in the state ? indicates that for this component, if ordinary chains with state space $1,\ldots,C$ were started in

every state at the same time in the past, then this component in every chain would not be the same. The summary states method is most easily described through an example.

Consider the attractive Ising model ($\beta > 0$, $C = 2$). In practice the Gibbs sampler simulating from the Ising model updating component (i, j) will use a $U(0, 1)$ variable γ, and the next state of component (i, j) of the chain will be x_{ij} where,

$$x_{ij} = \begin{cases} 0 & \text{if } \gamma \leq \mathbb{P}(x_{ij} = 0 | x_{-(ij)}) \\ 1 & \text{if } \gamma > \mathbb{P}(x_{ij} = 0 | x_{-(ij)}). \end{cases} \tag{4.7}$$

Due to the finite number of possible configurations of the cells neighbouring x_{ij} there will be a minimum value of $\mathbb{P}(x_{ij} = 0 | x_{-(ij)})$ over all possible configurations of the neighbouring cells. If γ is less than this minimum, then regardless of the state of the neighbouring cells x_{ij} will be 0. Similarly x_{ij} will be 1 regardless of the neighbouring states if γ is greater then the maximum value of $\mathbb{P}(x_{ij} = 0 | x_{-(ij)})$ over the possible configurations of the neighbourhood to x_{ij}.

The single chain followed in the summary states method starts off in a state with all components equal to ?. Then if the result of (4.7) is the same regardless of the possible values of any neighbouring cells with state ?, the next state of component (i, j) of the summary chain is given by x_{ij}, otherwise the component remains in state ?. Eventually every component of the summary chain will no longer be in the state ?. Once this happens there is no uncertainty in the next possible state and so the summary chain never returns to a state containing a ?. This will happen when all possible chains have coalesced. If the summary chain is run in a CFTP algorithm from sufficiently far in the past that no components remain in the state ? then the state at time 0 will be a sample from π.

This idea can clearly be extended to other choices of β and C in the obvious way, and indeed other distributions defined on a finite state space.

Childs et al. (2001) apply Huber's (1998) summary state method to the antiferromagnetic Potts model and show that the summary states method performs poorly at low temperatures by requiring to be run from an excessively long time in the past. The Gibbs sampler is known to suffer from a critical slowing down at a certain temperature, say T_G. However the number of iterations that the summary states method takes to converge increases sharply at a temperature above T_G. Childs et al. (2001) extend the summary states method to incorporate more information about the possible states at each update. They keep track of the subset of possible states of each cell, rather than just whether the state of the cell has been entirely determined. Childs et al. (2001) show that this offers an improvement over Huber's (1998) original method, but that

the number of iterations taken to converge still diverges at a temperature above T_G.

The mean number of iterations from 1600 runs taken by the summary states method to converge for the Potts model with $N = 64$ and $C = 3$ is shown in Figure 6 of Childs et al. (2001), which plots the convergence time for both Childs et al.'s (2001) and Huber's (1998) methods. It shows that the convergence time of Huber's (1998) original method increases from 10 to 100 iterations from a temperature of 5 down to 2.5, but at a temperatures of about 2.3 the time taken increases towards 1000 indicating that the convergence time diverges near a temperature of 2.2. The plotted mean convergence times of Childs et al.'s (2001) method show that at temperatures down to about 1.2 their method takes under 100 iterations to convergence, but that the convergence time increases rapidly at 1.1, and so also suffers from a divergent convergence time.

We now look at applying ACFTP to this model, to see whether ACFTP can be considered as a viable method for generating near perfect samples from this model for temperatures where the summary states method fails in practice because of its slow convergence at low temperatures.

4.8.4 Applying ACFTP

ACFTP is straightforward to apply to the Potts model using the Gibbs sampler defined by the conditional distributions given in (4.6). Each conditional distribution is easily generated from using one $U(0,1)$ random variate, in the same manner as described for the Ising example in Section 4.8.3. The discrete state space and the way that two chains are updated using one Uniform variate means that two coupled chains will coalesce naturally without any extra effort required. ACFTP is implemented in exactly the same manner as described in Chapter 3. We use the same criteria for assessing the performance of ACFTP that was used in previous sections; the proportion of chains starting from a sample from the stationary distribution is used as a measure of how well ACFTP does. To generate these samples from the stationary distribution, two long runs of a Gibbs sampler were used. Each run generated 8000 samples spaced 10000 iterations apart (with a burn-in of 10000 iterations). The long length of these runs should be sufficient to regard the 16000 samples as samples from π. As we shall see later from the results shown in Figure 4.22, in all the examples we look at chains should have forgotten their starting states after 10000 iterations, and so these 16000 samples can also be regarded as independent.

When ACFTP is implemented the user must find an over-dispersed distribution, and choose some parameters, m, r and s. From the previous sections, values of $m \geq 12$, $r \geq 4$ and $s \geq 3$ were recommended based on the examples looked at. We now look at using ACFTP with these values to sample from the antiferromagnetic Potts model

with $N = 64$ and $C = 3$ for various values of β.

For the Potts model it is not so clear how to choose a suitable starting distribution. The antiferromagnetic Potts model will have lot of modes based around checker board type patterns where each neighbouring pair are a different colour, this is called a q-colouring. The complexity of the Potts model means that there will also be other modes which consist of areas of checkerboard patterns. In previous examples ACFTP was seen to perform well even when the starting distribution was poorly chosen, such as when the starting distribution puts all its mass on one state. So here we look at fairly simple choices for the starting distribution.

One set of states which if included in the starting distribution should help the performance of ACFTP, by making sure that chains do not coalesce before they have reached modes, are states taking the form of q-colourings. There are a large number of states corresponding to q-colourings and sampling uniformly from these is a difficult problem in its own right. However generating particular q-colourings is straightforward. To incorporate an over-dispersed nature into the starting distribution we can include the states corresponding to lattices of all one colour, which will have the lowest probability under π. Of course there will be other states of very low probability which consists of large areas of the same colour, but we will see that our simple choice does well. Another choice of over-dispersed distribution could be achieved by using a Uniform distribution where each cell is independent and equally likely to be one of the 3 colours.

In the examples in the previous sections we saw that when implementing ACFTP a good choice of parameters is to take $m = 12$, $r = 4$ and $s = 3$. The 12 chains in each repetition is enough to be able to include several states from each of the three types of starting states that go into constructing the starting configuration. To obtain the results given in this section the starting configuration consisted of; 3 chains starting from each state of all one colour, 5 chains starting from Uniformly chosen states, and 4 chains starting from a randomly chosen checkerboard pattern. The random checkerboard patterns were obtained by starting in the top left corner of the lattice and working from left to right down the lattice, choosing the colour of a cell uniformly from the set of colours which do not match any of the colours of the cells neighbours.

We can anticipate the performance of ACFTP at different temperatures by looking at the mean forward coalescence times for chains starting from our starting configuration. In looking at these times we start 12 chains from the starting configuration and record how long they take to coalesce, which is then repeated 100 times. Figure 4.22 shows the mean coalescence time from runs at different temperatures. The temperature scale in Figure 4.22 corresponds to the temperature scale used in Figure 6 of Childs et al. (2001). It is clear that the Gibbs sampler still remains a viable method for simu-

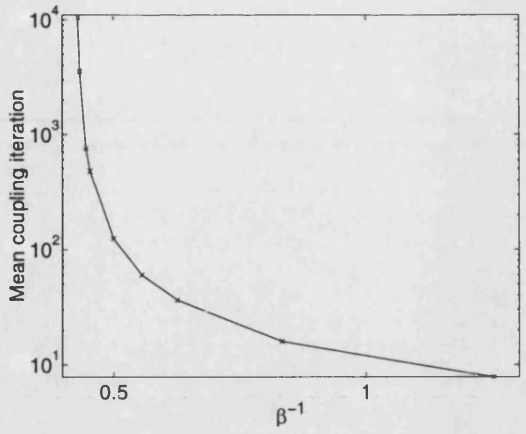

Figure 4.22: The mean coalescence time for 100 runs of 12 Gibbs sampler chains simulating from the Potts model started from the starting configuration for various temperatures.

temperature	0.43	0.45	0.5	0.56	0.83	1.25
proportion	0.99988	0.99938	0.99963	0.99963	0.99963	1.00000

Table 4.6: The proportion of stationary chains coalescing with the ACFTP chains for the Potts model with different temperatures, from 200 runs of ACFTP.

lating from the Potts model at temperatures below that at which the summary states methods suffer from critically slowing down; Huber's (1998) original method could not cope with temperatures below 2.2, and Childs et al.'s (2001) improvement still failed at temperatures below 1.1. Figure 4.22 shows that the Gibbs sampler remains a viable method for simulating from this Potts model at temperatures down to around 0.43. ACFTP should not suffer any critical slowing down other than that which the Gibbs sampler suffers from. So we now look at whether ACFTP returns states that are sufficiently close to the stationary distribution.

ACFTP was applied to the Potts model with $N = 64$, $C = 3$ using $m = 12$, $r = 4$ and $s = 3$, and the starting configuration described. Table 4.6 shows how the proportion of stationary chains coalescing with ACFTP chains varied across the temperature. For all the temperatures looked at, even 0.43, this proportion is very close to 1, showing that ACFTP is generating samples from close to π. The number of iterations in the past that ACFTP went back to was generally in line with the convergence times shown in Figure 4.22, tending to be the smallest time of the form 2^i greater than the forward coalescence time. This shows that ACFTP does indeed only slow down when the Gibbs sampler does.

4.9 Some Results from all the Examples

One thing which has not been mentioned so far when looking at ACFTP is its cost in terms of the number of iterations it uses. We first look at how many cycles of running chains from the past and collecting forward samples were typically required, that is, when implementing the ACFTP algorithm described in Section 3.2, typically how many times Steps 3 and 4 were cycled through. This is fairly similar across all the examples looked at in this thesis. Table 4.7 shows how many times forward samples were collected from 200 runs of ACFTP applied to each of the distributions looked at, using an over-dispersed starting distribution and with $m = 12$, $r = 5$ and $s = 3$. In all the examples at most two forward runs were required (except the Potts model which on one occasion needed three forward runs), and on no more than 11 out of 200 occasions did ACFTP need to get more than one set of forward samples. This shows how our very simple choices of over-dispersed starting distributions are good at providing the desired property of making ACFTP go sufficiently far back in the past.

The results from using poorer starting distributions reflect the quality of the starting distribution. For the point mass starting distributions, the number of forward sampling runs required depended on the position of the starting distribution. For the fixed dimension examples the state at which all initial chains were started was the mode (or a state very close to it), and ACFTP generally took 2 sets of forward samples on about 150 occasions with the other 50 occasions requiring only 3 collections of forward samples. The mixture distributions are multimodal and so the performance of ACFTP depends on which mode the single starting state was near. The mixture of regressions example followed a similar pattern to the fixed dimension examples, while the mixture of Normals example tended to require at least 3 forward sampling runs to finish. The results from using the stationary distribution as the starting distribution were more similar to those using the over-dispersed starting distribution. This demonstrates the benefits of using the forward samples as a check on how well the starting distribution covers the state space, and shows that ACFTP can still be reasonably efficient even if a poor starting distribution is chosen.

We have also not yet reported the number of iterations required overall by ACFTP. ACFTP is easily parallelized by running each repetition on a separate processor and so we consider the cost in terms of the number of iterations required by each processor (IP) to generate an independent starting state, where a chain will be started from each of these states in order to simulate states used in estimating properties of the desired distribution.

In the examples looked at IP was generally a small multiplicative factor greater

Distribution	Number of forward samples 1	2	3
MVN1	194	6	0
MVN2	196	4	0
MVN3	198	2	0
MVN4	196	4	0
pump	200	0	0
rats	200	0	0
mixture of Normals	193	7	0
mixture of regressions	192	8	0
Potts[4] with $\lvert\beta^{-1}\rvert = 0.45$	189	10	1

Table 4.7: The number of forward sampling runs used by ACFTP to return r samples from various distributions using $m = 12$, $r = 5$ and $s = 3$, out of 200 runs.

than the time that ACFTP went back into the past, which we denote by $-T < 0$. For example across all of the distributions looked at, when using values of $m = 12$, $r = 4$, and $s = 3$, ACFTP tends to result in a value of IP roughly between $15T$ and $25T$. We feel that these figures are perfectly acceptable when compared to perfect simulation algorithms.

The best we could expect to take would be that needed by the simplest version of CFTP which is Propp & Wilson's (1996) original algorithm that follows only two chains, the minimum and maximum chains, and for this we would expect IP to be around $3T$ iterations[5]. More complicated perfect simulation algorithms will take longer, for example, the algorithms of Green & Murdoch (1999) which partition the state space will run many chains from the past starting from states in the different partitions, and dominated CFTP (Kendall & Møller 2000) uses chains that are run both into and from the past. So comparatively ACFTP is not too costly.

4.10 Discussion of ACFTP

In this chapter we have applied the algorithm of ACFTP, for obtaining almost perfect samples, to a range of example distributions arising in Bayesian statistics for which perfect sampling algorithms are not available. The Bayesian models considered included

[4]Results for the Potts model used $r = 4$.

[5]To get these figures we note that both of the the minimum and maximum chains will be run for $2 + 4 + 8 + \cdots + \frac{T}{2} = T - 2 \approx T$ iterations in total during the runs from the past when the chains do not coalesce. Then the two chains are finally run from time $-T$ when they do coalesce and we might expect them to take $0.5T$ iterations to do so. The total number of iterations that the two chains will likely take is therefore around 3 times T.

two examples of mixture distributions to which even regular convergence diagnostics cannot be applied. ACFTP was also implemented on a discrete example from statistical physics, where although a perfect simulation algorithm exists, the perfect simulation algorithm becomes impractical for certain choices of a parameter in the model. Simulated tempering was also looked at, in order to allow the Potts model to be simulated from at lower temperatures, however the results are not shown in this thesis. The extra variable describing the current temperature of a chain performing simulated tempering presented no difficulties for ACFTP. In all of these examples, with a suitable choice of input parameters, ACFTP was shown to do well at generating samples from close to the stationary distribution. Although these form only a small set of examples, it is hoped that they serve as an indicator of how ACFTP could be applied to a wider range of distributions, both continuous and discrete.

ACFTP has the advantage over perfect simulation algorithms of having more flexibility in the choice of updates which can be used, allowing it to be used with a wider range of distributions. When comparing it to diagnostic methods based on the output of a chain it was shown to broadly agree on the number of iterations required for a chain to converge, showing that the extra effort needed to coalesce chains did not require the chains to be run for a lot more iterations. In terms of the assessment of convergence, ACFTP diagnoses convergence when multiple chains coalesce to the same state, and so this provides an indicator of convergence based on all the parameters converging as a whole. Diagnostics that base their results on functions of the chain's output may diagnose convergence prematurely because although these functions of the chain's state have converged the chain as a whole has not, which is the view taken by Brooks & Gelman (1998) who suggest the MPSRF.

We have also seen that ACFTP can be applied to some distributions that existing diagnostic methods cannot. Such a class of distributions are those where the dimension of the parameters can vary, and we applied ACFTP to some mixture distributions with this property. The methods that have been proposed for assessing convergence of mixture distributions assess convergence based on a subset of the parameters common to all models, and this may suffer from diagnosing convergence too early. ACFTP was shown to successfully return samples from close to the stationary distribution in all the examples looked at. ACFTP also has the benefit of generating r independent samples which can be used to run r independent chains and this will provide better estimates of variances.

We can also consider the computational and programming cost. If parallel processors are available then ACFTP can be easily programmed to run on them, and so the total running time may be reduced. The extra programming required to implement

ACFTP need only be written once, as the methods for coupling and coalescing chains can be written as black box routines. So we feel that there is not a great burden in using ACFTP.

The performance of ACFTP was shown to be easily measurable by checking coalescence with stationary chains. This approach is one which can be used with other diagnostics that use the coalescence of chains to diagnose convergence. Two examples of schemes using coalescence are the following.

Johnson (1996) derives an analytical bound on the total variation distance between the distribution of a chain at time t and the stationary distribution based on the probability of two chains not coalescing. His method uses repeated runs of chains to assess the distribution of the iteration at which chains coalesce to obtain this bound. Further simulation work looking at particular examples could be used to give an idea of how far away the bounds are in practice.

Neal (2002) describes a method for simulating from a distribution by running a chain in a circular fashion. The number of desired samples N from the distribution must be chosen in advance. A chain is started at time 0 and run circularly using the same random deviates at each time in its updates, so the state at time N becomes the state at time 0. If a coalescing step is included every so often, eventually the chain should cycle through the same sequence of states for $t = 0, \ldots, N$. Other chains are started at times between 0 and N from an over-dispersed distribution and checked to see how quickly they coalesce with the main chain. If the checking chains all coalesce satisfactorily then the algorithm stops and returns the N samples as samples from the stationary distribution. If not then Neal (2002) suggests using a larger value of N and repeating the procedure, however he does not mention whether new random variates would be used. Neal's (2002) algorithm is similar to ACFTP in the way that he uses checking chains started from over-dispersed starting states to check coalescence. The problem of choosing N in Neal's (2002) method means that the implementation is trickier especially because of any bias issues that might arise when the algorithm fails with the first attempted choice of N, which is not addressed in Neal's (2002) paper. ACFTP does not have the problem of choosing N because it burns in a chain, and it also does not come with any stopping time bias.

We envisage that ACFTP serves as a substitute for perfect simulation algorithms when they are not available to sample from the desired distribution. Some of the methods used by perfect simulation algorithms can be incorporated into ACFTP to coalescence chains, but because ACFTP is not a perfect simulation algorithm it is much more flexible in the methods it can use to update chains. ACFTP can also be used along side regular diagnostic methods to provide extra assurances about the

simulation results, and can be used when convergence diagnostics cannot be applied to a problem, as in the case of mixture distributions.

4.11 Further Work

Clearly ACFTP has its limitations. The requirement that chains coalesce means that at the moment ACFTP is restricted to use with problems where there are methods available for updating coupled chains that lead to them getting closer together, so that they can be coalesced using simple methods. Getting two chains to coalesce is a problem for all perfect simulation algorithms. Overcoming this problem for ACFTP should be easier because it does not require that chains starting from all possible states coalesce to the same state; for example the problem of distributions on unbounded state spaces and coping with the infinite tails of the distribution are avoided. To construct an updating method where chains get closer together; convenient conditional distributions might be achieved through the introduction of auxiliary variables; or a particular type of updating method could be used, for example Neal (2002) looks at using Langevin updates to get coupled chains closer together. We have investigated the coalescence properties of other algorithms which are commonly used to update chains.

Adaptive rejection sampling (ARS), Gilks & Wild (1992), is a popular algorithm used to generate samples from log-concave distributions. From looking at one example, the coalescence properties seem to depend on the implementation of ARS and the choice of initial starting abscissae. The more independent the ARS algorithm is of the previous state of the chain the more likely it will be to coalesce chains. If ARS uses information from the previous state of the chain to place abscissae then the chains do not tend to get closer together and so cannot be coalesced easily. We have looked at coalescing chains by restricting the rejection envelopes to use points on a coarse grid, which should mean that two similar conditional distributions use the same envelope over some parts of the state space. However in order to gets chains to coalesce it required the grid to be very coarse. The more coarse the grid the further away from the conditional distribution the envelope is and this leads to more samples from the envelope distribution being generated which is inefficient. This is an area we could consider for future investigation.

Looking at every possible method for updating a chain and seeing whether it brings chains closer together does not seem the sensible approach. It is better to look for flexible algorithms that have good convergence and coalescence properties which can be used with any distribution. An example might be slice sampling. In Chapter 2 we saw that the slice sampler has many appealing features, such as good convergence,

monotonicity properties, and the possibility to coalesce chains directly. The slice sampler is a flexible method that could be applied to a very large range of distributions hopefully without too much effort, and something to look at in the future could be whether slice samplers tend to bring coupled chains closer together.

Chapter 5

Can the Problem of Masking Outliers in Bayesian Linear Models be Overcome using Mode Jumping Methods in MCMC?

5.1 Introduction

The problem of detecting outliers in Bayesian models is a difficult task and attention has generally been restricted to linear models. Many different approaches have been proposed of which a few are now briefly mentioned. Model elaboration has been suggested by Box & Tiao (1968) and Guttman (1973), who incorporate a mechanism for the generation of outliers in a linear model. Box & Tiao (1968) suggest modelling the outliers as having a larger variance than the rest of the observations, while Guttman (1973) models outliers by a shift in their expected value. Other suggested approaches involve using the predictive distribution to assess the probability of subsets of observations arising from a fitted model (Box 1980, Geisser 1980, Pettit 1990). In this chapter we look at using Box & Tiao's (1968) variance inflation model, which has proved to be popular because of the convenience of identifying outliers through the model using an MCMC approach.

Verdinelli & Wasserman (1991) show how estimates of the probability of an observation being an outlier can be obtained with this model using the Gibbs sampler. However Justel & Peña (1996) show that the Gibbs sampler may fail to detect outliers that mask one another, which is a problem common to any method that tries to detect outliers. Justel & Peña (2001) describe a procedure to overcome the problem of iden-

tifying masked outliers, however it does not attempt to correctly assess the probability of subsets of masking outliers outlying. In this chapter we look at whether a variation of the mode jumping method described by Tjelmeland & Hegstad (2001) can be used to better assess this probability.

5.2 The Normal Linear Model

We consider the Bayesian regression model where observations $y = (y_1, \ldots, y_n)'$ are generated by

$$y = X\beta + \varepsilon$$

where X is an $n \times p$ known matrix of full rank, β is a $p \times 1$ vector of parameters and ε is an $n \times 1$ vector of independent Normal errors with mean 0 and variance σ^2. Later on we will be applying this model to an example taken from Justel & Peña (2001), so we consider this model using the same improper prior used by them, which is to take

$$p(\beta, \sigma^2) \propto \frac{1}{\sigma^2} .$$

Of course other choices are possible, for example a multivariate Normal prior for β and an inverse Gamma prior on σ^2.

5.3 Variance Inflation Model for Outliers

Box & Tiao (1968) expanded the model defined in Section 5.2 to include a mechanism for the generation of outliers. They assume that each observation is either an outlier, with probability α, or not an outlier, with probability $(1 - \alpha)$. If observation i is an outlier then Box & Tiao (1968) assume that $\varepsilon_i \sim N(0, k^2\sigma^2)$ and if not then $\varepsilon_i \sim N(0, \sigma^2)$. For fixed values of α and k they show that the posterior distribution of β is a weighted average of 2^n t-distributions, with each distribution corresponding to each possible subset of the observations labelled as outliers. Box & Tiao (1968) give a formula for obtaining the relative weight associated with each possible subset of the observations being the outliers. These weights are only available up to a normalising constant, so calculating all 2^n weights is impractical. Box & Tiao (1968) recommend assuming that there are at most only a few outliers and only including the corresponding distributions in the mixture, with the weights appropriately renormalised. This still leaves the question of how to identify the outliers in the first place.

For later convenience we now define the following notation. If A is an $n \times m$ matrix and I is a subset of $\{1, \ldots, n\}$ then let A_I denote the matrix containing the rows whose

indices are given by the set I. Let $\bar{I}$ denote the compliment of I and $|I|$ the number of elements in I. For parameters and distributions obtained under the assumption of a certain set of the outliers we follow the convention of using the subscript (I) when the set I is assumed to be the set of indices of the outliers.

Verdinelli & Wasserman (1991) show how the Gibbs sampler may be used to identify outliers and assess their probability of outlyingness. This overcomes the problem in Box & Tiao's (1968) original paper which requires that the possible subsets of outliers are prespecified before the model is fitted. The flexibility of using the Gibbs sampler also means that models where α varies can be easily considered which is much more difficult using the analytical approach of Box & Tiao (1968). Verdinelli & Wasserman (1991) easily incorporate this into the Normal linear model by putting a Beta(γ_1, γ_2) prior on α, where they take $\gamma_1 + \gamma_2 = n$. This will be chosen to reflect the assumption that the proportion of outliers is small. Verdinelli & Wasserman (1991) show how the set of parameters $(\beta, \sigma^2, \alpha)$ can be augmented with the parameter vector $\delta = (\delta_1, \ldots, \delta_n)$ where if I indicates the current set of outliers then

$$\delta_i = \begin{cases} 1 & \text{if } i \in I \\ 0 & \text{if } i \notin I\,, \end{cases}$$

and at each iteration every observation has a certain probability of being an outlier based on the current values of $(\beta, \sigma^2, \alpha)$. Using the improper prior for (β, σ^2) and the Beta prior for α, with a fixed value of k the conditional distributions of each parameter are available in closed form. For $V = diag(1 + \delta_i(k^2 - 1))$, $\hat{\beta} = (X^T V^{-1} X)^{-1} X^T V^{-1} y$, $\phi = 1 - \frac{1}{k^2}$, $r = \sum_i \delta_i$ and $\nu = n - p$, the conditional distributions are

$$\beta | \sigma^2, \alpha, \delta \sim N_p(\hat{\beta}, \sigma^2 (X^T V^{-1} X)^{-1}) \tag{5.1}$$

$$\sigma^{-2} | \beta, \alpha, \delta \sim \Gamma(\frac{n}{2}, \frac{1}{2}(y - X\hat{\beta})^T V^{-1} (y - X\hat{\beta})) \tag{5.2}$$

$$\alpha | \beta, \sigma^2, \delta \sim Beta(\gamma_1 + r, \gamma_2 + n - r) \tag{5.3}$$

$$\mathbb{P}(\delta_i = 1 | \beta, \sigma^2, \alpha) = \left(1 + \frac{1 - \alpha}{\alpha} k \exp\left\{ -\frac{\phi(y - x_i^T \hat{\beta})}{2\sigma^2} \right\}\right)^{-1}. \tag{5.4}$$

A Gibbs sampler can then be run using these conditional distributions, and the probability of observation i being an outlier can be estimated using the proportion of iterations that $\delta_i = 1$. However as Justel & Peña (1996) point out the posterior distribution arising from this model can be multimodal, and the Gibbs sampler can take a long time to move between different modes. There could be two subsets of observations, I and J, such that the Gibbs sampler takes a very long time to move between a state

where the subset I of observations are labelled as outliers, and a state where the subset J are labelled as outliers. Justel & Peña (1996) give examples of this happening with masking outliers, that is where there are subsets of observations that do not outlie individually, but do outlie when every observation in the subset is considered to be an outlier. It may also be that there is more than one set of plausible outliers under the model, leading to a multimodal posterior distribution.

Justel & Peña (1996) suggest a method for finding any subsets of masked outliers using repeated runs of the Gibbs sampler. Any non-masking outliers should be picked up in a straightforward way by the Gibbs sampler so in describing their method only subsets of outliers that are masking are assumed to exist. For the purposes of Justel & Peña's (1996) algorithm they consider a situation in which the observations consist of a set $\bar{J}$ indicating "good" observations which are actual realizations from the Normal linear model, and the set of remaining observations J indicating "bad" observations which do not arise from the model and mask one another. The purpose of their method is then to correctly identify the good and bad observations. Justel & Peña (1996) surmise that with such a set of observations there will be two modes when fitting a variance inflation model. The first mode will correspond to correctly identifying the good and bad observations. They suppose that in the second mode the bad observations will mask one another, resulting in some good observations that are incorrectly labelled as outliers and the bad observations incorrectly labelled as non-outliers. The mislabelled good observations are referred to as swamped.

The Gibbs sampler will be slow to move between these two modes because of the strong influence of the masking outliers, and so the outliers may fail be identified. Justel & Peña (1996) observe that if the Gibbs sampler is repeatedly run starting in a randomly chosen state where most of the observations are labelled as non-outliers and only a few are labelled as outliers, then it will converge randomly to one of the two modes. The mode it converges to will depend on the initial state of the Gibbs sampler. However because the Gibbs sampler does not move freely between the two modes then it is not possible to tell whether the bad observations are correctly labelled as outliers.

Justel & Peña (2001) suggest repeatedly running the Gibbs sampler starting from states that contain only a few randomly chosen observations labelled as non-outliers, and monitoring the value of δ at the end of each run. Based on the assumed structure of the modes, they suppose that for two masking outliers i and j the correlation between δ_i and δ_j should be high, because they will only tend to outlie jointly. They also suppose that the correlation of the allocation parameters between two non-outliers is small and that the correlation between non-outliers and outliers is also small. Justel & Peña's (2001) method for identifying the masking outliers is then; for each run of R runs of

the Gibbs sampler, record the value of δ at the end of each run, and then estimate the covariance matrix of the allocation vector based on these R samples. Justel & Peña (2001) show how, under the assumptions about the covariance properties of the good and bad observations, the eigenvalues and eigenvectors of the estimated covariance matrix can be used to indicate a subset of the observations that should be free of bad observations. It then remains to find out which of the rest of the observations are the swamped good data and which are the masking outliers. When starting from a state containing a large number of correctly identified good observations and with the rest of the observations labelled as outliers, the Gibbs sampler should converge to the mode where the bad observations are labelled as outliers, and therefore identify the masking outliers. Justel & Peña (2001) indicate that this argument can be extended to incorporate more than one set of masking outliers.

For a full Bayesian analysis of the data, Justel & Peña (1996) suggest that the probability of each observation being an outlier can now be assessed by using the output of the Gibbs sampler when it is started from what is believed to be the correct mode. However doing so does not truly assess the probability of the labelled outliers outlying under the model, because the Gibbs sampler does not mix throughout the state space, and we will see an example of this later. We now go on to look at whether the mode jumping algorithm of Tjelmeland & Hegstad (2001) could be used to move between and find the different modes, which should give a better reflection of the true probability of subsets of the observations outlying.

5.4 Mode Jumping Proposals using Local Optimizations in the Metropolis-Hastings Algorithm

Tjelmeland & Hegstad (2001) show how optimizations that find local maxima can be incorporated into a Metropolis-Hastings step which finds well separated modes of the target distribution π and allows jump between them. They note that the standard Metropolis-Hastings algorithm on $\mathbb{R}^p$ can be generalized in the following two ways.

The first is that if we have a set of transition kernels $\{P^\varphi\}^{\varphi\in\mathbb{R}^d}$ each satisfying detailed balance with respect to the stationary distribution π, and have a density $f(\varphi)$ on $\mathbb{R}^d$ then the transition kernel given by

$$P(A|x) = \int_{\mathbb{R}^d} P^\varphi(A|x) f(\varphi) d\varphi \tag{5.5}$$

also satisfies detailed balance.

The second generalization assumes that we have two proposal distributions on $\mathbb{R}^p$,

$q_0(\cdot|\cdot)$ and $q_1(\cdot|\cdot)$. One way of combining these two proposal densities is to choose between them at each iteration. If the current state of the chain is x, randomly choose one of the proposal distributions, then propose a new state y that is generated from the chosen proposal, $q_i(\cdot|\cdot)$ say. The proposed next state y is then accepted with probability

$$\alpha_i(y|x) = \min\left\{1, \frac{\pi(y)q_i(x|y)}{\pi(x)q_i(y|x)}\right\}.$$

Tjelmeland & Hegstad (2001) observe that this type of transition can be generalized to give more flexibility in using the proposal densities q_0 and q_1. If with probability $\frac{1}{2}$ y is sampled from $q_i(\cdot|x)$, then an alternative move that satisfies detailed balance accepts y with probability

$$\alpha_{i,1-i}(y|x) = \min\left\{1, \frac{\pi(y)q_{1-i}(x|y)}{\pi(x)q_i(y|x)}\right\}. \tag{5.6}$$

The move that has this acceptance probability consists of first choosing between the two proposals q_0 and q_1 each with probability $\frac{1}{2}$. Without loss of generality we assume that q_0 has been chosen. The proposed next state y is then obtained by generating a sample from $q_0(\cdot|x)$. This proposal is then matched with q_1, so that the probability of proposing the reverse move from y to x is $q_1(x|y)$.

The fact that this type of move satisfies detailed balance can be seen by checking it directly using the the transition kernel, which is given by,

$$P(A|x) = \frac{1}{2}\int_A q_0(y|x)\alpha_{0,1}(y|x)\mathrm{d}y + \frac{1}{2}\int_A q_1(y|x)\alpha_{1,0}(y|x)\mathrm{d}y + I[x \in A]r(x) \tag{5.7}$$

where $r(x)$ is the probability of rejecting the proposed state,

$$r(x) = \frac{1}{2}\left(\int_{\mathbb{R}^p} q_0(y|x)(1-\alpha_{0,1}(y|x))\mathrm{d}y + \int_{\mathbb{R}^p} q_1(y|x)(1-\alpha_{1,0}(y|x))\mathrm{d}y\right).$$

This transition kernel will satisfy detailed balance if,

$$\pi(x)P(y|x) = \pi(y)P(x|y).$$

Substituting in (5.7), and removing the rejection probabilities which cancel directly, gives,

$$\begin{aligned}
&\pi(x)q_0(y|x)\min\left\{1, \frac{\pi(y)q_1(x|y)}{\pi(x)q_0(y|x)}\right\} + \pi(x)q_1(y|x)\min\left\{1, \frac{\pi(y)q_0(x|y)}{\pi(x)q_1(y|x)}\right\} \\
&= \pi(y)q_0(x|y)\min\left\{1, \frac{\pi(x)q_1(y|x)}{\pi(y)q_0(x|y)}\right\} + \pi(y)q_1(x|y)\min\left\{1, \frac{\pi(x)q_0(y|x)}{\pi(y)q_1(x|y)}\right\}.
\end{aligned}$$

Both sides are equal (the first and second terms from either side cancel) and thus detailed balance holds for a chain with the transition kernel given by (5.7).

Tjelmeland & Hegstad (2001) combine these two generalizations of the Metropolis-Hastings algorithm to produce a mode jumping step. At each iteration a transition kernel P^{φ} is chosen with probability $f(\varphi)$. This transition kernel uses two proposal densities q_0^{φ} and q_1^{φ} and then the proposed new state y, generated from proposal q_i^{φ} with probability $\frac{1}{2}$, is accepted with probability

$$\alpha_{i,1-i}^{\varphi}(y|x) = \min\left\{1, \frac{\pi(y)q_{1-i}^{\varphi}(x|y)}{\pi(x)q_i^{\varphi}(y|x)}\right\}. \tag{5.8}$$

Both proposals q_0^{φ} and q_1^{φ} are generated in the same way, so we first consider a way to construct q_0^{φ} which will allow the chain to jump to a new mode. The construction of q_0^{φ} consists of three parts and this is illustrated in Figure 5.1[1]. We assume that the chain has already converged to a local mode but that it does not ordinarily move between different modes of π. Tjelmeland & Hegstad (2001) suggest choosing f to be a distribution on $\mathbb{R}^p$ that is over-dispersed relative to the possible location of any modes, so here $d = p$. Then, using a sample φ from the distribution $f(\cdot)$; from the current state x the first part of the mode jump is to make a move from x to $T_0(x, \varphi) = x + \varphi$, which should serve as a jump away from the current mode. Next the nearest mode is located by a deterministic local maximization started at $T_0(x, \varphi)$. Letting $\mu(x)$ denote the location of the mode when the maximization is started at x, the maximization returns $\mu(T_0(x, \varphi))$. A suitable density centred near $\mu(T_0(x, \varphi))$ can then be chosen as the proposal. This density should ideally be slightly over-dispersed relative to π at this local mode. Tjelmeland & Hegstad (2001) suggest using

$$q_0^{\varphi}(\cdot|x) = N_p(\mu(T_0(x, \varphi)), \Sigma(T_0(x, \varphi)))(\cdot)$$

where $N_p(\mu, \Sigma)(\cdot)$ denotes the density function of a p-variate Gaussian distribution with mean μ and covariance matrix Σ, and $\Sigma(T_0(x, \varphi))$ returns a suitable covariance matrix based on both the states visited by the optimization algorithm and/or any known properties of the target distribution. The proposed next state y is then generated from this proposal.

In order to try and achieve a high probability of acceptance q_1^{φ} should be chosen so that there is a high probability of the chain jumping from y back to x. One way to try and achieve this is to reverse the jump away from the mode used in q_0^{φ}, which is illustrated in Figure 5.1. Tjelmeland & Hegstad (2001) suggest taking an initial jump

[1] Figure 5.1 is based on a diagram that appeared in Tjelmeland & Hegstad (2001).

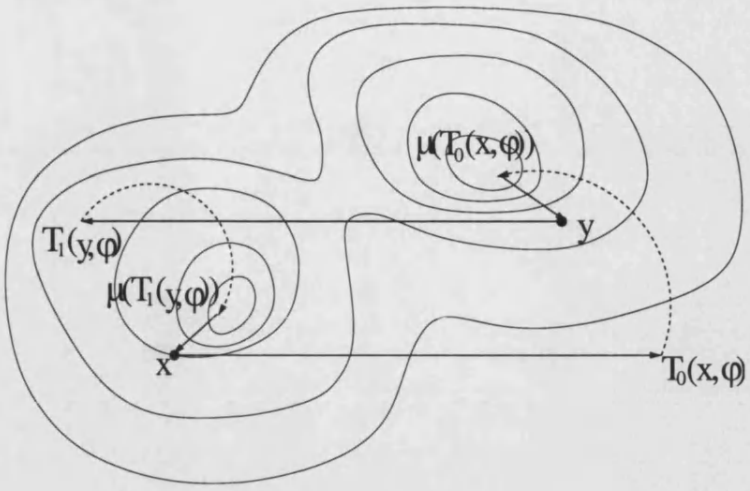

Figure 5.1: Diagram illustrating Tjelmeland & Hegstad's mode jumping algorithm.

away from y to $T_1(y,\varphi) = y - \varphi$ where φ is the same realization of the random variable used in constructing q_0^{φ}. The jump to $T_1(y,\varphi)$ should hopefully then be back in the direction of x and lie in the basin of attraction of the mode that x is in. The local optimization routine can be used to find the nearest mode, $\mu(T_1(y,\varphi))$ and then the return proposal is given by

$$q_1^{\varphi}(\cdot|y) = N_p(\mu(T_1(y,\varphi)), \Sigma(T_1(y,\varphi)))(\cdot) \ .$$

If $\mu(T_1(y,\varphi))$ is the location of the mode that x lies in, then the probability of returning to x under this proposal should be high. In such a situation the acceptance probability $\alpha_{0,1}^{\varphi}$ will also therefore be high, and the chain will have a good chance of jumping between the widely separated states x and y.

In the next section we show how Tjelmeland & Hegstad's (2001) mode jumping algorithm may be viewed as an auxiliary variable method, and how it may be extended so that the proposal distributions q_0 and q_1 can be centred at a point determined by a sequence of random jumps from the current state. In Section 5.6 we look at using a sequence of random jumps to locate the nearest mode, so that the difficulties of programming a suitable optimization routine can be avoided. Any optimization routine should make use of any local knowledge of π, however an algorithm which does this and finds local modes is already used in constructing the chain. Using the mode jumping step at every iteration is clearly inefficient, (as mentioned by Tjelmeland & Hegstad (2001)), and so the mode jumping move is only used every so often in combination with a simpler straightforward algorithm that makes local moves, for example the Gibbs

sampler. If the local updating method is started in a state away from a mode then it will converge toward one, so we could consider using this as our local optimization. The program code for generating samples using the local updating method will conveniently already exist, and so we extend Tjelmeland & Hegstad's (2001) method by replacing the optimization with iterations of the standard updating method for the chain. We then go on to look at applying the new mode jumping algorithm to the problem of identifying masking outliers in the variance inflation model.

5.5 The Mode Jumping Algorithm as an Auxiliary Variable Method

Tjelmeland & Hegstad (2001) use the fact that their mode jumping move is a combination of the Metropolis-Hastings algorithms described by (5.5) and (5.8) to state that their algorithm satisfies detailed balance. Both individual algorithms satisfy detailed balance and therefore so does a combination of them. In this section we show that both Tjelmeland & Hegstad's (2001) method and the extended mode-jumping method which will be looked at in Section 5.6 may be viewed as auxiliary variable methods.

In both Tjelmeland & Hegstad's (2001) Metropolis-Hastings move and the extended version in Section 5.6 the proposed next state is generated by making a sequence of random moves from the current state, the last state of which is the proposed next state. The only probability distribution used in making the sequence of random moves to appear directly in the acceptance probability is the last one in this sequence. For example in Tjelmeland & Hegstad's (2001) method the distribution f does not appear in their acceptance probability, given by (5.8), even though it is used to make a random jump away from the current state before the proposed next state is sampled. To achieve this we show how an auxiliary variable Φ can be introduced which allows the chain to make a sequence of random moves, before making the final proposal which does appear in the acceptance probability.

Let X be a random variable defined on the state space $\mathcal{X} = \mathbb{R}^p$ with distribution π_X. Introduce an auxiliary variable Φ with state space $\Omega = \mathbb{R}^d$ such that the joint density of (X, Φ) is given by

$$\pi_{X\Phi}(x, \varphi) = \pi_X(x) f(\varphi)$$

where f is an easy to sample from density on Ω. Clearly X and Φ are independent and so the marginal distribution of X is π_X, and the conditional distributions are simply

given by the marginal distributions,

$$\pi_{\Phi|X}(\varphi|x) = f(\varphi) \tag{5.9}$$

and

$$\pi_{X|\Phi}(x|\varphi) = \pi_X(x) \ . \tag{5.10}$$

A Markov chain using these conditional distributions will generate samples from the distribution $\pi_{X\Phi}$ and the realizations of the variable X can then be taken as samples from the distribution π_X. Using these conditional distributions a Metropolis-Hastings chain can be constructed where the method of proposing the next state of X consists of making a sequence of random jumps which are not incorporated into the acceptance probability, followed by the final proposal which is. We now describe how this may be achieved.

The variable Φ is updated first, by sampling directly from f. Let (x, φ) be the state of the chain having just updated the state of Φ.

The X variable is updated by using a Metropolis-Hastings step using two proposal distributions $q_0(\cdot|\theta(x,\varphi))$ and $q_1(\cdot|\theta(x,\varphi))$; both of these proposal distributions are defined by a set of parameters θ, for example they could be Normal distributions which each have a mean and a variance, so $\theta = (\mu, \sigma^2)$. In most cases the parameters in θ will depend on the value of x and φ. These two proposals are combined in the manner described in Section 5.4 which leads to an acceptance probability of the form (5.6). That is with probability $\frac{1}{2}$ proposal q_i is chosen and a proposed new state y generated from this proposal. The state y is then accepted as the next state of the chain with probability

$$\alpha_{i,1-i}(y|x) = \min\left\{ 1 \ , \ \frac{\pi_X(y) q_{1-i}(x|\theta(y,\varphi))}{\pi_X(x) q_i(y|\theta(x,\varphi))} \right\} \ . \tag{5.11}$$

Random jumps used in making the proposals that do not have to be incorporated into the acceptance step can be generated through the construction of θ, which defines a function describing how the parameters of the proposal distributions depend on the current state of the chain. We now describe a very simple example of how this might be achieved, which although not a useful method itself should make the idea clearer.

As a straightforward example we construct a move where the proposed next state is generated by first making a Normal random walk Metropolis-Hastings move (with variance s^2), and then making another Normal random walk Metropolis-Hastings move centred at the state obtained by the previous move (with variance σ^2). The proposed next state of the X component is the value sampled from the second proposal, and the first move does not directly appear in the acceptance probability for the proposed next

state.

For this simple example we take $\mathcal{X} = \mathbb{R}$ and $\Omega = \mathbb{R}^4$, so $\varphi = (\varphi_1, \ldots, \varphi_4)$. The two Normal proposals $q_0(\cdot|\theta_0)$ and $q_1(\cdot|\theta_1)$, where $\theta_0 = (\mu_0, \sigma^2)$ and $\theta_1 = (\mu_1, \sigma^2)$, will be constructed in the same way, so for the purpose of describing the move we assume that the proposal away from the current state is made using q_0, which we construct first. Let (x, φ) be the current state of the chain. The first move in generating the proposed next state for the X component will consist of a Normal proposal centred at the current state with fixed variance s^2. This proposal is made using one Normal variable and one Uniform variable, so we take $\varphi_1 \sim N(0, s^2)$ and $\varphi_2 \sim U(0,1)$ independently. The first Metropolis-Hastings move is defined as, move to $x + \varphi_1$ if,

$$\varphi_2 < \min\left\{1, \frac{\pi_X(x+\varphi_1)}{\pi_X(x)}\right\}$$

otherwise remain at x. The second Metropolis-Hastings move which does appear in the acceptance probability has a Normal proposal with mean μ_0 (which depends on x and φ), and fixed variance σ^2. The mean of this proposal, μ_0, is taken as the state at which the first Metropolis-Hastings step finished. This can be achieved by choosing

$$\begin{aligned}\mu_0(x,\varphi) = (x+\varphi_1) I\left[\varphi_2 < \min\left\{1, \frac{\pi_X(x+\varphi_1)}{\pi_X(x)}\right\}\right] \\ + x I\left[\varphi_2 \geq \min\left\{1, \frac{\pi_X(x+\varphi_1)}{\pi_X(x)}\right\}\right] .\end{aligned} \tag{5.12}$$

The proposed next state for the X component is then sampled from a $N(\mu_0, \sigma^2)$ distribution, the density of which we denote by $q_0(\cdot|\theta_0(x,\varphi))$. The construction of q_1 follows in exactly the same way, except that the independent variables $\varphi_3 \sim N(0, s^2)$ and $\varphi_4 \sim U(0,1)$ are used instead of φ_1 and φ_2. The proposed next state y is then accepted with probability,

$$\min\left\{1, \frac{\pi_X(y) q_1(x|\theta_1(y,\varphi))}{\pi_X(x) q_0(y|\theta_0(x,\varphi))}\right\} ,$$

otherwise the X component of the chain remains in state x. In this simple example only one Metropolis-Hastings step is made before the final proposal but it can clearly be extended to include more steps, and indeed any other type of random moves, not necessarily Metropolis-Hastings steps or even steps defining a Markov chain. If many random moves are used in a row then each μ_i is the composition of each of the individual moves. The number of moves itself can be random (depend on φ) with the further use of indicator functions. In the next section we will see that the explicit formula for each

μ_i does not need to be calculated, as the result of it can be generated by applying the random moves in turn.

The acceptance probability formula given in (5.11) corresponds to the equivalent formula used by Tjelmeland & Hegstad (2001) given in (5.8). In Tjelmeland & Hegstad's (2001) algorithm they deal with a set of proposals q^{φ} indexed by the variable φ, whereas we use the fact that this is equivalent to having one proposal whose set of parameters depend on the variable φ.

Hopefully the auxiliary variable view of the Metropolis-Hastings mode jumping method makes it easier to see how random moves can be made before a proposed next state is generated, where the random moves do not appear in the acceptance probability.

We now consider ways to show that our chain has the desired stationary distribution and satisfies appropriate balance conditions. The auxiliary variable method produces a chain with stationary distribution $\pi_{X\Phi}(x,\varphi)$ whose transition kernel satisfies general balance, because the transition kernels for both the X and Φ steps satisfy detailed balance when considered on their own; the Φ step is simply a Gibbs sampling move and the X step is a Metropolis-Hastings move, which both satisfy detailed balance.

A chain which satisfies detailed balance as a whole can be obtained by integrating out the variable Φ at each iteration. The resulting chain consists of purely an X component. The transition kernel for this chain is given by,

$$P(A|x) = \int_A \int_{\mathbb{R}^d} \sum_{i=0}^{1} \frac{1}{2} f(\varphi) q_i(y|\theta(x,\varphi)) \alpha_{i,1-i}(y|x) + f(\varphi) I[x \in A] r(x) \mathrm{d}\varphi \mathrm{d}y,$$

where $\alpha_{i,1-i}$ is taken from (5.11) and $r(x)$ is the probability of rejecting the proposal,

$$r(x) = \frac{1}{2} \left(\int_{\mathbb{R}^p} q_0(y|\theta(x,\varphi))(1 - \alpha_{0,1}(y|x)) \mathrm{d}y + \int_{\mathbb{R}^p} q_1(y|\theta(x,\varphi)(1 - \alpha_{1,0}(y|x)) \mathrm{d}y \right).$$

The condition of detailed balance requires that $\pi_X(x)P(y|x) = \pi_X(y)P(x|y)$. This holds because for values of $x \neq y$,

$$\begin{aligned}
\pi_X(x)P(y|x) &= \pi_X(x) \int_{\mathbb{R}^d} \sum_{i=0}^{1} \frac{1}{2} f(\varphi) q_i(y|\theta(x,\varphi)) \alpha_{i,1-i}(y|x) \mathrm{d}\varphi \\
&= \int_{\mathbb{R}^d} \sum_{i=0}^{1} \frac{1}{2} f(\varphi) \min\{\pi_X(x) q_i(y|\theta(x,\varphi)), \pi_X(y) q_{1-i}(x|\theta(y,\varphi))\} \mathrm{d}\varphi \\
&= \pi_X(y) \int_{\mathbb{R}^d} \sum_{i=0}^{1} \frac{1}{2} f(\varphi) q_{1-i}(x|\theta(y,\varphi)) \alpha_{1-i,i}(x|y) \mathrm{d}\varphi \\
&= \pi_X(y) P(x|y),
\end{aligned}$$

and is clearly true when $x = y$.

However to show that a chain which makes a sequence of random jumps before making the final proposal satisfies detailed balance, requires no more than the arguments given in Tjelmeland & Hegstad (2001), which we now show. Just as Tjelmeland & Hegstad (2001) do, let φ denote the move type, which determines the sequence of random moves before the proposed next state is made. This means that given φ, every starting state x has a destination state $\varphi(x)$, so we have a map defined by $x \to \varphi(x)$. Once φ is known this map is deterministic. The map can take any form desired. To obtain the proposed next state we generate y from a density we can write down directly, given by $f_0^{\varphi(x)}(\cdot)$ say. We can now suppress the mechanism of these two stages by saying that given φ we have a well-defined proposal when in state x denoted by $q_0^{\varphi}(\cdot|x)$. The return proposal from y to x is produced in the same way, by combining the map defined by $y \to \varphi(y)$ and the return proposal $f_1^{\varphi(y)}(\cdot)$ into one well-defined proposal given by $q_1^{\varphi}(\cdot|y)$. Choosing between proposals of type 0 and type 1 with equal probability will result in the acceptance probability used by Tjelmeland & Hegstad (2001), shown in (5.8).

In the next section we use this new formulation for the mode jumping move so that random moves can be included in the optimization step of Tjelmeland & Hegstad's (2001) algorithm.

5.6 Extending the Mode Jumping Metropolis-Hastings Algorithm Further

When considering Tjelmeland & Hegstad's (2001) mode jumping algorithm, as described in Section 5.4, as a special case of the auxiliary variable method described in Section 5.5 there is clearly a lot of freedom allowed in the choice of Φ, $f(\cdot)$ and the random moves made in the Metropolis-Hastings step before the final proposal. We now look at choosing them so that the sample from f can be used to both jump away from the current mode and to locate the nearest mode after the jump away. To do this we will let φ consist of four random vectors by taking,

$$\varphi = (\xi_{M0}, \xi_{M1}, \varphi_0, \varphi_1)'$$

where each vector is a sequence of random variables which are all independently generated. The construction of our Metropolis-Hastings move will take the same form as that of Tjelmeland & Hegstad (2001).

From the current state (x, φ), using the proposal q_0, the first move will be a jump

away from the current mode made using the function T_0. The function T_0 makes this random jump using ξ_{M0}. This is then followed by a sequence of "random moves" defined by the function $\mu_0(T_0(x, \xi_{M0}), \varphi_0)$, which is a deterministic function of its arguments but where the randomness is produced by φ_0, just like the example in Section 5.5. The state returned after T_0 and μ_0 have been run is then used by q_0 to generate the proposed next state. For example the mean parameter in the distribution q_0 might be the state returned by μ_0. The return proposal q_1 is constructed in exactly the same way but using ξ_{M1} and φ_1.

The method for jumping away from the current mode (that is the choice of each T_i) will depend on particular properties of the distribution considered. Therefore we leave looking at the move used to make a large jump away from the current mode until we apply the mode jumping to the variance inflation model. We note that in this implementation of the mode jumping each T_i uses a different independent set of random variables to make the jump away from the current mode. This will be unlikely to work well in problems with many modes, but worked fine on the examples with a few modes looked at in this thesis. In general when implementing the jump away from the current mode, the T_i should use the same random variable ξ. The T_i should be chosen so that there is a high probability of the return move reaching the mode where the current state is. For example this can be done by constructing the T_i so that $T_0(T_1(x, \xi), \xi) = T_1(T_0(x, \xi), \xi) = x$, such as the way described in Section 5.4.

The complex nature of the mode jumping steps mean that it is clearly sensible that they are not used on their own, but in conjunction with local moves. The local moves will act like a stochastic mode finding algorithm, because chains started in a state away from a mode will tend to converge towards one. It is possible to describe the process of making these local updates through the function Ψ in a stochastic recursive sequence (SRS), as described in Section 1.6.

The mode finding, that is each μ_i, can therefore be the function defined by the composition of every Ψ used to define a finite run of a SRS, in an extension of the simple example in Section 5.5. The vector of random variates in φ_i will be chosen to contain the appropriate random variates needed by the SRS. The explicit formula for μ_i need not be calculated however, as the result of $\mu_i(x, \varphi)$ for a particular (x, φ) can be calculated by running the SRS.

The update function Ψ can consist of any type of random moves generated using the variates in φ. The choice of Ψ is not just restricted to Metropolis-Hastings moves as described in the simple example in Section 5.5, or even to updates from a Markov chain. The only problem that might arise is if the function Ψ generated the random steps from a rejection algorithm, so here each $\varphi_0^{(t)}$ would consist of a long finite sequence of

random variables. In this case each μ_i would consist of lots of combinations of indicator functions containing all the possible results of the accept/reject steps using the variates in φ_0. The only difference between a proper rejection sampler and the rejection sampler which would be used here is that here the rejection sampler might reach the end of the sequence of $\varphi_0^{(t)}$, in which case the outcome will have to be defined. This is not a problem however because the sequence $\varphi_0^{(t)}$ can be chosen to be of an arbitrarily long finite length as it need only be generated as required.

Thus the local optimization can now be replaced by a finite run of a SRS simulating from π using a given sequence of random variates. For convenience the proposal q_0^{φ} will use φ_0 to find a local mode and q_1^{φ} will use φ_1. The same set of random variates could be used, but this avoids the need to keep track of seeds which would require extra programming.

5.7 Mode Jumping in the Variance Inflation Model

The mode jumping will be used to improve mixing in linear regression problems where there are masking outliers. In this situation we assume that there are well separated modes corresponding to different subsets of observations labelled as the outliers. We assume that the chain has been running long enough that it has converged to one of the modes of π and that we now wish to make a step which is likely to propose a move to a different mode. The construction of a Metropolis-Hastings move to do this will follow a similar design to that of Tjelmeland & Hegstad (2001). It will use the Metropolis-Hastings transition kernel described in Section 5.6 to jump between modes and use the Gibbs sampler defined in (5.1)-(5.4) to update the chain most of the time by making local moves. The proposed move is formed from three parts, a large initial jump away from the current mode, a climb toward a (hopefully new) mode, and a proposal away from the new mode.

We now discuss how the jump away from the current mode is constructed. Let $\theta_t = (\beta_t, \sigma_t^2, \alpha_t, \delta_t)$ be the current state of a Markov chain simulating from the posterior distribution of the variance inflation model which we denote by $\pi(\theta)$. Justel & Peña (2001) point out that for a mode which consists of observations I that are generally labelled as outliers and the rest non-outliers, if the Gibbs sampler is started in a state where δ contains mostly 1s except for a few $\delta_i = 0$ where $i \notin I$ then it will tend to converge toward the mode. Therefore one suggestion for jumping away from the current mode could be to attempt to swap the outliers and non-outliers. We could do this by using the following method.

Let J denote the indices of the current set of outliers, so for the current value of δ

we have,

$$\delta_j = \begin{cases} 1 & \text{if } j \in J \\ 0 & \text{if } j \in \bar{J} \end{cases} . \tag{5.13}$$

In order to jump away from the current mode, we jump to a state with only a few non-outliers, where these non-outliers come from the list of currently labelled outliers. The climb to the nearest mode should hopefully then find a different mode. The jump away from the current mode is achieved by choosing a small value $n_0 \geq p$ which will be the number of non-outliers, and then

if $|J| \geq n_0$
    choose $j_1, \ldots, j_{n_0}$ uniformly from J
otherwise
    choose $j_1, \ldots, j_{n_0}$ uniformly from $\{1, \ldots, n\}$.

The jump away from the current mode is then to a state with allocation variables $\delta_{(\mathrm{F}_0)} = (\delta_{(\mathrm{F}_0)1}, \ldots, \delta_{(\mathrm{F}_0)n})$ where

$$\delta_{(\mathrm{F}_0)i} = \begin{cases} 0 & \text{if } i \in \{j_1, \ldots, j_{n_0}\} \\ 1 & \text{otherwise} \end{cases} .$$

The allocation parameter $\delta_{(\mathrm{F}_0)}$ will then consist of a small number of zeros assigned to observations that were previously labelled as outliers. In order to complete the jump away from the current mode, values need to be chosen for the other parameters β, σ^2 and α. This is discussed later in the context of the hill climbing.

In constructing the mode jumping move and the mode climbing part we will have to be careful about the use of the components of φ. Each hill climb consists of a SRS forming a Gibbs sampler. The Gibbs sampler requires a stream of random variables in order to update a chain at each iteration. The form of this stream of random variables will depend on the specific construction of the algorithm used to generate samples from the conditional distributions. Samples from the same conditional distribution will be able to be generated in many ways. For some distributions it may be possible to generate samples from them using a transformation of a standard variable, such as a Normal distribution, which then only require a fixed number of random variates as input. Other schemes could involve a rejection part which potentially requires an unbounded number of random variates. Each component of φ consists of a finite number of random variables fixed in advance, and so we will look at one way to deal with this through the example of the variance inflation model, but there are others ways to cope with this that extend to more complicated examples, and these will be discussed at the

end of the chapter. One method of coping with the fixed number of random variables is to make sure there is a maximum number that will be needed. For the mode jump described above this is easily achieved by using at most n $U(0,1)$ variables, let ξ_{M0} be such a vector of n $U(0,1)$ variables.

The mode finding part consists of running a SRS forming a Gibbs sampler using the conditional distributions in (5.1)-(5.4), so we now see how φ_0 and φ_1 can be suitably chosen for use with such a SRS. It is possible to generate a sample from each of these distributions using a fixed number of random variables. The following methods for doing so were taken from Devroye (1986).

A sample from the multivariate Normal distribution in (5.1) can be generated using $\xi_\beta \sim N_p(0, I_p)$ and a lower triangular matrix H which satisfies $HH' = \sigma^2(X^TV^{-1}X)^{-1}$, by returning $\hat{\beta} + H\xi_\beta$.

To update σ^{-2} according to its Gamma conditional distribution, a random variable $\xi_{\sigma^{-2}} \sim \Gamma(\frac{n}{2}, 1)$ is used and the next state is taken as $\xi_{\sigma^{-2}}/(\frac{1}{2}(y - X\beta)^TV^{-1}(y - X\beta))$, which is a sample from (5.2).

One way to sample from the Beta conditional distribution of α, which although inefficient uses a fixed number of random variates independent of the current state, is by generating $\xi_{\alpha(1)}, \ldots, \xi_{\alpha(2n-1)} \sim U(0,1)$, and then if $\xi_{\alpha(i)}$ is the $(\gamma_1 + r)$th largest value, we update α to $\xi_{\alpha(i)}$ because this is then a draw from a $Beta(\gamma_1 + r, \gamma_2 + n - r)$ distribution. In the examples looked at here the parameters of the conditional distribution are only ever integers and so this method will always be able to be used.

Each component i of δ is easily updated using 1 $U(0,1)$ variable ξ_{δ_i} by comparing it to the expression in (5.4).

The SRS generating samples from each conditional distribution therefore only requires a finite number of random variates. If we specify the maximum number of iterations for which the Gibbs sampler will be run in order to find the nearest mode, T_M say, then we can take

$$\varphi_0 = (\xi_\beta^{(1)}, \xi_{\sigma^{-2}}^{(1)}, \xi_{\alpha(1)}^{(1)}, \ldots, \xi_{\alpha(2n-1)}^{(1)}, \xi_{\delta_1}^{(1)}, \ldots, \xi_{\delta_n}^{(1)}, \xi_\beta^{(2)}, \xi_{\sigma^{-2}}^{(2)}, \ldots, \xi_{\delta_n}^{(T_M)})$$

where the superscripts refer to the iteration of the SRS used to find a mode. In order to find a mode we could run a SRS using φ_0 until it has either converged to a mode according to some convergence diagnostic, or it has run for T_M iterations. In practice what we actually do is run two different kinds of stochastic recursive sequences one after another for a fixed number of iterations, in an effort to avoid having the SRS be too affected by any masking outliers when the number of non-outliers is small. Even though the effect of the outliers is downweighted through the conditional distributions

they still affect the regression line, with only a few non-outliers this effect can be significant enough to move the regression line to fit any masking outliers.

We would like the hill climbing algorithm to converge towards a mode where the regression line closely fits the non-outliers in $\delta_{(\mathrm{F}_0)}$ and those observations close by. To do so we run M_1 iterations of the following algorithm. We update the components $(\sigma^2, \alpha, \delta)$ using the conditional distributions (5.2)-(5.4) and the methods requiring a finite number of random variates previously described. To update β we use the distribution

$$\beta|\sigma^2, \alpha, \delta \sim N_p(\hat{\beta}_\delta, \sigma^2(X^T V^{-1} X)^{-1})$$

where $\hat{\beta}_\delta = (X_{\bar{K}}^T X_{\bar{K}})^{-1} X_{\bar{K}} y$ and K is defined by

$$\delta_i = \begin{cases} 1 & \text{if } i \in K \\ 0 & \text{if } i \notin K \end{cases} .$$

Updating β in this way reduces the influence of the outliers because $\hat{\beta}_\delta$ has no dependence on the value of the outlying observations, (it is the least squares estimate with the observations labelled as outliers by the current state of δ removed from the dataset). If the components are updated in the order β, σ^{-2}, α, δ then when the SRS starts after the mode jump, only δ and σ^2 need to be specified in order to start off the cycle of generating samples from (5.1)-(5.4). If the mode climbing SRS is started with $\sigma^2 = 0.001$ then it should mean that the fitted line follows the observations labelled as non-outliers in $\delta_{(\mathrm{F}_0)}$. Once this algorithm has finished the state it returns should be a sample from close to a local mode. It will prove useful later when we construct the proposal within this mode to obtain a sample from the mode. To do this we run a few iterations of a SRS with the desired stationary distribution, starting from the final state of the previous SRS. So this second SRS consists of M_1 iterations of the conditional distributions in (5.1)-(5.4). Let the subscript (F_2) denote the state of any parameters at the end of this SRS.

In the same manner as Tjelmeland & Hegstad (2001) we then make a proposal away from $\theta_{(\mathrm{F}_2)}$. Tjelmeland & Hegstad (2001) mention the problem of how to generate a proposed new state for a chain with a mixture of continuous and discrete parameters. They suggest using a Uniform proposal for the discrete parts and then using their mode jumping method to generate a proposal for the continuous parameters. However the distribution of the discrete components is likely to be far from Uniform and this choice will likely result in low acceptance probabilities for the mode jumping moves. To overcome this problem in the variance inflation model we use one cycle of the distributions (5.1)-(5.4) to generate a proposed next state θ'. If the Gibbs sampler

mixes reasonably well within each mode and $\theta_{(F_2)}$ is a representative sample from the current mode, then the proposal should have a good chance of acceptance. Let $q_0^{\varphi}(\theta'|\theta)$ denote this proposal. For the return step we would like a high probability of proposing a move from the proposed new state to the current state of the chain. The return step is constructed in the same way as the forward proposal, that is a jump away from the current mode, a hill climb and then a proposal within the new found mode. The variates ξ_{M1} and φ_1 are generated in the same way as ξ_{M0} and φ_0. The jump away from the current mode, and the two algorithms used to climb the nearest hill are implemented in the same way as well. The proposal probability is then the probability of generating θ via one cycle of the conditional distributions (5.1)-(5.4) when starting in the state returned by the second SRS used on the return step. The probability of the return jump is denoted by $q_1^{\varphi}(\theta|\theta')$. After the return move has been calculated we then accept θ' as the next state of the chain with probability

$$\alpha_{01}(\theta'|\theta) = \min\left\{1, \frac{\pi(\theta')q_1^{\varphi}(\theta|\theta')}{\pi(\theta)q_0^{\varphi}(\theta'|\theta)}\right\},$$

otherwise the chain remains in state θ.

This describes the mode jumping step if we were to use q_0 to propose a new state and q_1 as the reverse move. We do this with probability 1/2, but also with probability 1/2, q_1 will be used first. In this particular example q_0 and q_1 are constructed in exactly the same way, so there is no real difference in the choice of which proposal is used to go forward.

We now investigate the performance of this mode jumping step when simulating from the variance inflation model using a dataset with known masking outliers.

5.8 Examples

Justel & Peña (2001) apply their algorithm for unmasking outliers in linear models to a number of established examples of datasets containing masking observations, and it is one of these examples that we investigate using the mode jumping method from Section 5.7. However we first look at a simple example to demonstrate the assumed properties of the ordinary Gibbs sampler. For both of the examples we follow Justel & Peña's (2001) specification and use $k = 10$.

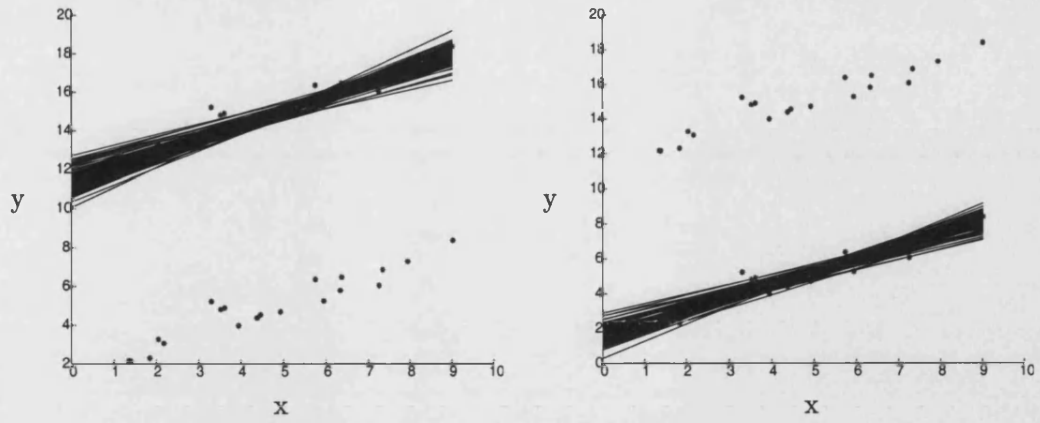

Figure 5.2: Fitted lines from 10^5 iterations of the Gibbs sampler simulating from the two lines example, starting in different states.

5.8.1 Two Lines

Consider a data set made up of $n = 40$ data following two straight lines where

$$y_i = \begin{cases} 2 + 0.6x_i + \varepsilon_i & i = 1, \ldots 20 \\ 10 + y_{i-20} & i = 21, \ldots 40 \end{cases}$$

for some values of x_i in the interval $(1, 9)$ and where each $\varepsilon_i \sim N(0, 0.7^2)$.

If a linear model is fitted to this data it is clear that the posterior distribution will be multi-modal and that the Gibbs sampler on its own will not move easily between modes. This is demonstrated by Figure 5.2 which show the fitted lines for one run of the Gibbs sampler for 100000 iterations and a burn-in of 500. To produce the fitted lines on the left in Figure 5.2 the Gibbs sampler was started with

$$\delta = \delta_0 = (0, 1, \ldots 1, 0)^T$$

and the plot on the right of Figure 5.2 was obtained by starting with

$$\delta = \delta_1 = (\underbrace{0, \ldots 0}_{20 values}, \underbrace{1, \ldots 1}_{20 values})^T.$$

Figure 5.2 shows that the Gibbs sampler does not move freely around the state space, and that it converges towards one of these two modes. A particular mode is easily found by starting the Gibbs sampler in a state where δ consists of mostly ones (outliers) except for a few zeros (non-outliers), where the non-outliers agree with the mode.

This distribution should however be straightforward to simulate from using the

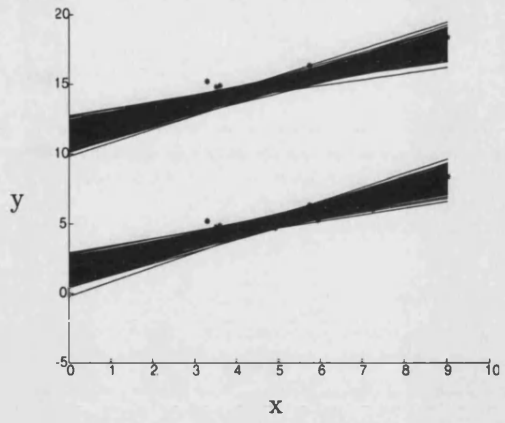

Figure 5.3: Fitted lines from runs of the mode jumping algorithm applied to the two lines example.

mode jumping from Section 5.7. As previously mentioned mode jumping steps should not be used alone, but in combination with other moves that perform more efficient local updates. For this example 9 iterations of the Gibbs sampler are used and then 1 Metropolis-Hastings mode jumping move. We take $n_0 = 3$, $M_0 = 25$ and $M_1 = 25$ as defined in Section 5.7. The algorithm was run for 1000 iterations and the fitted lines from a particular run are shown in Figure 5.3. It is clear that the algorithm moves freely between modes as desired. The acceptance probability for the mode jumping moves in this run was 0.48.

5.8.2 Rousseeuw Data Set

Justel & Peña (2001) apply their algorithm for unmasking outliers to an artificially constructed dataset from Rousseeuw (1984). Rousseeuw (1984) used this dataset to demonstrate his method of least median of squares regression for robustly fitting a regression line, for which other methods fail to identify the outliers. The data consist of 30 "good" observations generated according to

$$y_i \sim N(a + bx_i, 0.2) \quad i = 1, \ldots 30$$

with $a = 1$, $b = 2$ and the x_i are uniformly distributed on the interval $(1, 4)$, and 20 "bad" observations generated from a bivariate Normal distribution centred at $(7, 2)$ and with covariance matrix $0.5^2 I_2$. The particular set of observations used in this thesis are plotted in Figure 5.4. To investigate the modes of the posterior distribution of the variance inflation model applied to the Rousseeuw data, multiple runs of the

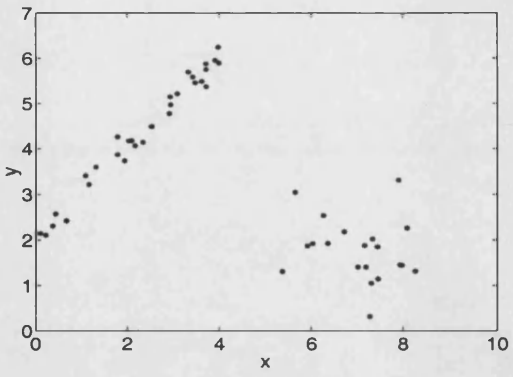

Figure 5.4: The Rousseeuw data set.

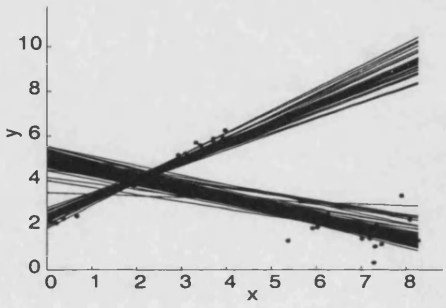

Figure 5.5: Fitted lines from the last iteration of 500 Gibbs sampler runs simulating from Rousseeuw example.

Gibbs sampler were used starting from the state $(\beta, \sigma^2, \alpha, \delta) = ((0,0), 1, \alpha_0, \delta)$ with $\sum_i (1 - \delta_i) = 3$ where the values of $\delta_i = 0$ are uniformly chosen and $\alpha_0 \sim Beta(3, 47)$. This choice should allow separate runs of the Gibbs sampler to reach different modes. The Gibbs sampler was run 500 times for 2000 iterations, which is more than sufficient for convergence to a local mode. The fitted lines from the last iteration of the 500 Gibbs sampler runs are shown in Figure 5.5 clearly demonstrating the multimodality of the posterior distribution. It was also observed that in none of the runs does the Gibbs sampler move between the two modes corresponding to $\beta_2 > 0$ (call this mode 1) and $\beta_2 < 0$ (mode 2), although it may move between smaller modes within each of these.

The mode jumping algorithm described in Section 5.7 was used to simulate from this posterior distribution. However it was found that mode jumps from mode 1 to

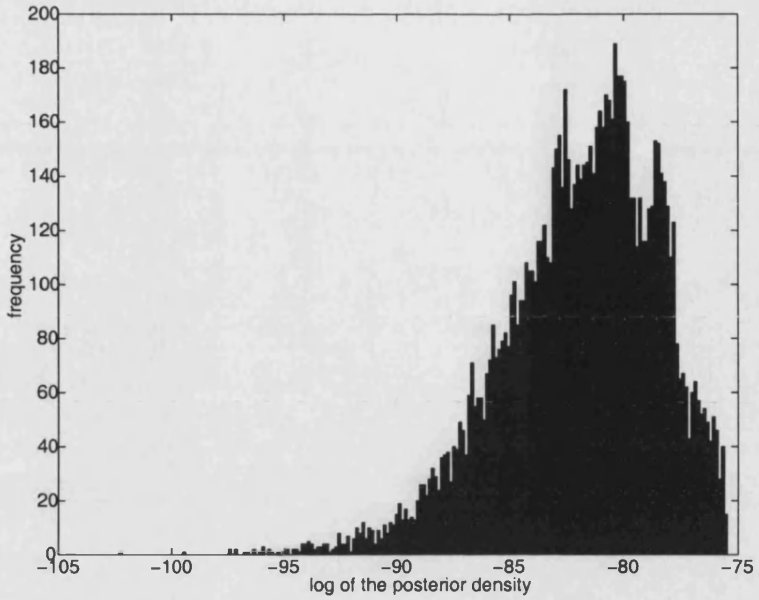

Figure 5.6: Log posterior density when the Gibbs sampler is simulating from mode 1.

mode 2 were almost always accepted whereas jumps from mode 2 to mode 1 were hardly ever accepted. This is not a fault of the mode jumping method, but correctly reflects the posterior distribution. To show that this is the case, we look at histograms of the log of the posterior density for 100 Gibbs sampler runs each of 100 iterations, after a suitable burn-in, started in modes 1 and 2. Figure 5.6 shows a histogram of the posterior density at each iteration of the Gibbs sampler started in mode 1 and Figure 5.7 shows the equivalent results when the Gibbs sampler is started in mode 2. Due to the fact that the Gibbs sampler does not move freely between the modes indicated in Figures 5.6 and 5.7, these plots do not indicate the weight associated with each mode. It is now clear why the mode jumps are rarely accepted. In the acceptance probability of a mode jump we have the ratio

$$\frac{\pi(\theta')}{\pi(\theta)}. \tag{5.14}$$

where θ is the current state of the chain and θ' is the proposed next state. When proposing a jump from the right mode in Figure 5.7 to the mode with the next highest posterior density values in Figure 5.6 this ratio is likely to be smaller than $\exp\{-77 - (-73)\} = 0.018$ and most of the time significantly smaller. Therefore once the chain has made its way to the right mode in Figure 5.7 it will tend to stay there for a large number of iterations and the rare excursions will be short lived.

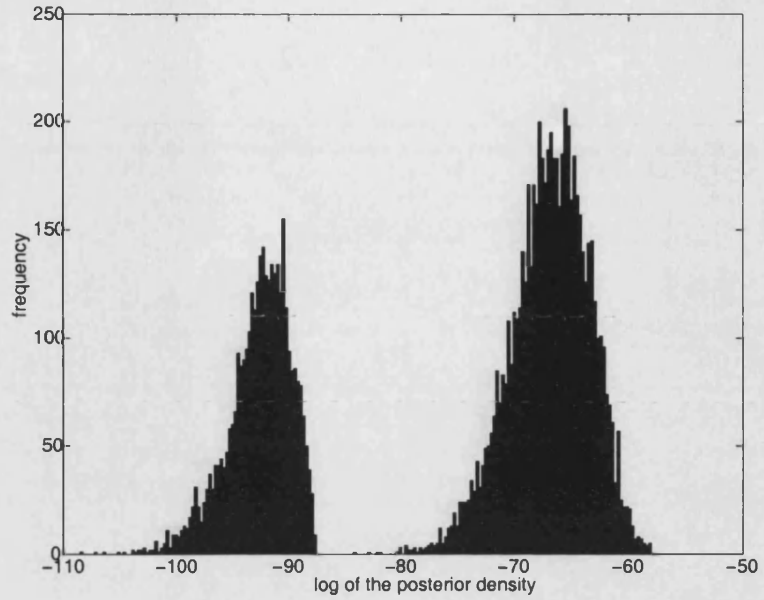

Figure 5.7: Log posterior density when the Gibbs sampler is simulating from mode 2.

The sampler could be made to mix between the modes by running the sampler on a different model where the set of states corresponding to each mode have been re-weighted so that the sampler moves between the modes. The probabilities of the sampled values could then be corrected by importance sampling. However we have not looked at this in this thesis, instead in Section 5.8.2 the sampler is implemented on an example data set with more equal modes.

It seems that under the variance inflation model the most likely conclusion is that the fitted line has a negative slope. The data clearly have a different structure to that which they would have if they were generated from the variance inflation model. Any inference about the probability of outlyingness for the "bad" observations based on fitting a line through the "good" observations is therefore misleading. Justel & Peña's (2001) method identifies the fact that there is more than one mode, but fails to obtain their relative weights according to the model. The mode jumping algorithm identifies the fact that the modes have very different weights through the ratio (5.14) appearing in the acceptance probability. It might also be suggested that the variance inflation model is possibly not appropriate for inference about the Rousseeuw data, although it is still useful for identifying outliers.

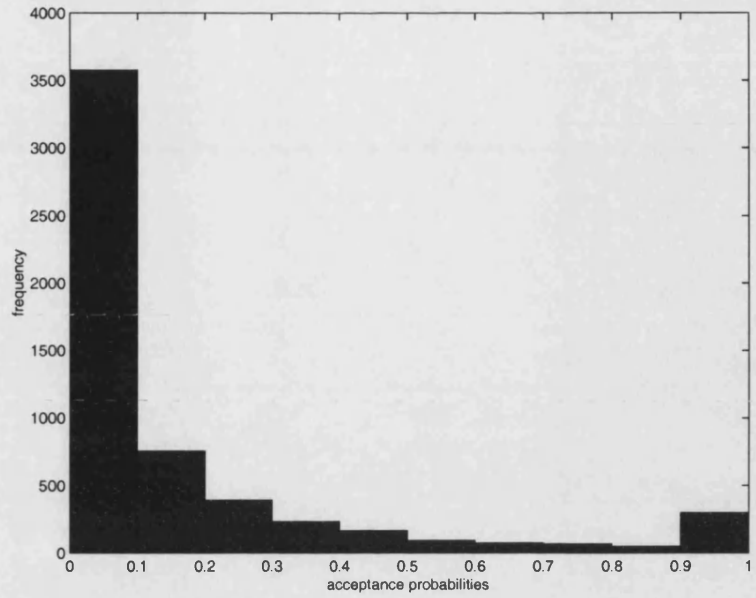

Figure 5.8: Histogram of the frequency of acceptance probabilities for attempted mode jumps from mode 1 to mode 2.

Reduced Rousseeuw Data Set

To demonstrate that the mode jumping algorithm can move between significant modes, it is used to simulate from the variance inflation model applied to data with the same structure as the Rousseeuw data but with fewer bad observations. We now consider a data set with 30 good observations and 12 bad observations. The Gibbs sampler still does not move between the modes of $\beta_2 > 0$ (mode 1) and $\beta < 0$ (mode 2). We construct a chain to sample from this distribution by running a standard Gibbs sampler but every tenth iteration a mode jump is attempted. We use $M_0 = 50$ and $M_1 = 50$.

It was noticed that that the acceptance probabilities for the mode jumping can be noticeably improved by using a more dispersed distribution for the proposal for β. We replace the multivariate Normal proposal for β with the p-dimensional multivariate t-distribution

$$\beta|\sigma^2, \alpha, \delta \sim t_p(\hat{\beta}_\delta, \sigma^2(X^T V^{-1} X)^{-1}, n-p).$$

A t-distribution was chosen because the marginal distribution of β when considering a model with a fixed value of α is a t-distribution, (Box & Tiao 1968).

This chain was run 500 times each for 200 iterations, after a suitable burn-in starting from a state with 3 uniformly chosen non-outliers. In all there were 10000 attempted mode jumps, of which 5712 were from mode 1 to mode 2, and 3351 were from mode 2

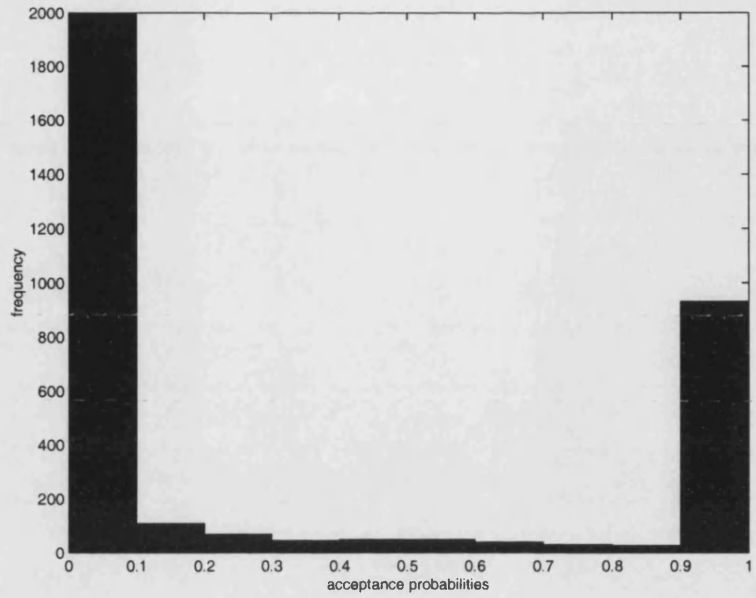

Figure 5.9: Histogram of the frequency of acceptance probabilities for attempted mode jumps from mode 2 to mode 1.

to mode 1, with the rest being made to the same mode. Figure 5.8 shows a histogram of the acceptance probabilities for attempted jumps from mode 1 to mode 2, and Figure 5.9 shows this for attempted jumps from mode 2 to mode 1. The mode jumping steps clearly have reasonable acceptance rates, in fact the overall acceptance rate for mode jumping attempts was 0.21. This conclusion is clarified by the multimodal nature of the estimated posterior densities shown in Figure 5.10, and the fitted lines at the final state of each Gibbs sampler run, Figure 5.11.

5.9 Conclusions

One of the disadvantages of the mode jumping algorithm of Tjelmeland & Hegstad (2001) is the use of specialist optimization routines which may be computationally expensive and which will require extra programming. In this chapter we have seen how the burden on a user of Tjelmeland & Hegstad's (2001) method could be reduced, by allowing the user to make use of existing computer code to construct the hill climbing algorithm. The use of φ in the hill climbing also allows the user more flexibility in their choice of hill climbing algorithm. One example could be where there are small local modes which are not of interest, Tjelmeland & Hegstad's (2001) mode jumping algorithm could get stuck in one of these modes because of the small chance of returning

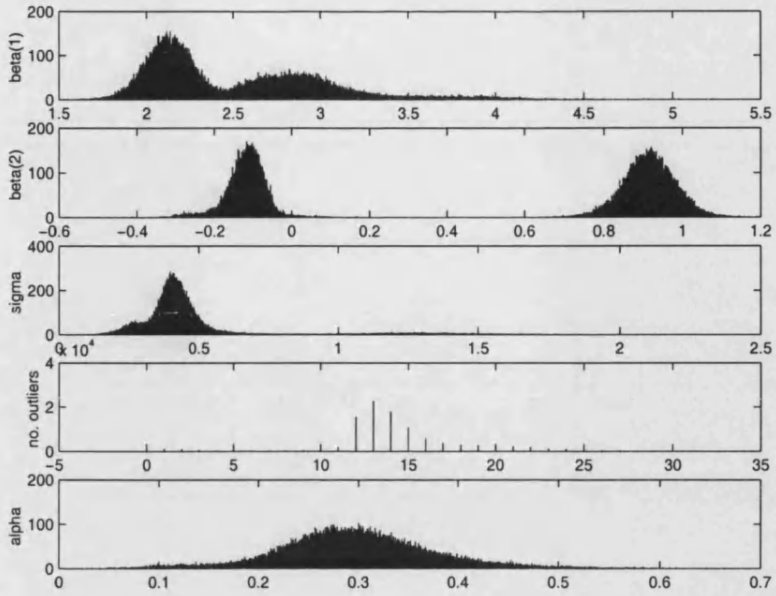

Figure 5.10: Histogram of simulated parameter values for the reduced Rousseeuw data set.

to it. To overcome this simulated annealing could be used to escape a local mode, and make use of the random variates in φ.

The restriction on the use of component updates which use a finite number of random variables in the mode climbing is easily removed. If the Gibbs sampler had used a rejection algorithm for generating from one of its conditional distributions then the hill climb can be stopped if it runs out of random variables to use. The number of random variables included in φ can be made sufficiently large so that this situation should not arise.

Through the use of the mode jumping chain described in Section 5.7 we have also demonstrated the need for care when making inference about multimodal distributions based on chains which do not mix well. If a single Gibbs sampler simulating from the variance inflation model was applied to the Rousseeuw dataset, even when starting from what is identified to be the correct mode the posterior estimates of outlying observations will be misleading.

The mode jumping algorithm was successful in its application to the Rousseeuw dataset, however the particular implementation described in Section 5.7 would probably not be so efficient at simulating from large datasets with many subsets of outlying observations that were hard to find. Constructing the jump away from current mode so that there is a good chance of jumping back to the same mode under the return

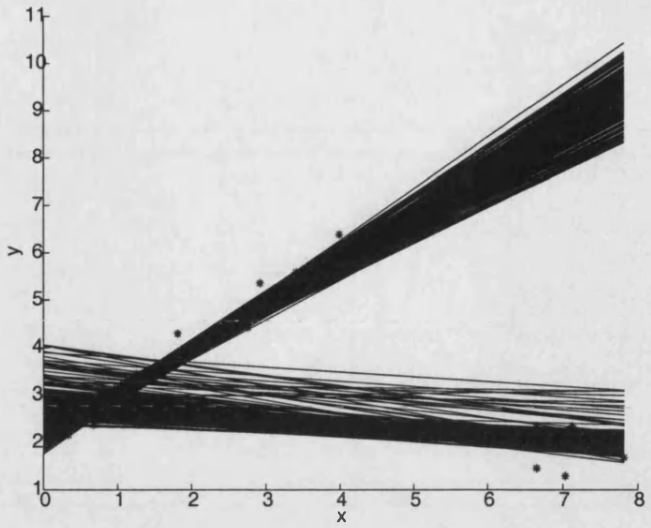

Figure 5.11: Fitted lines at the final state of each of the 500 Gibbs sampler runs for the reduced Rousseeuw dataset.

route proved to be difficult if there were several modes. One of the problems noted at the end of Tjelmeland & Hegstad (2001) is in constructing a suitable proposal for the discrete parts of a proposal away from a found mode. For the Rousseeuw dataset one cycle of the Gibbs sampler was seen to be sufficient in order to generate an efficient proposal, and we would expect this to be reasonably successful when used with larger datasets.

One thing we have not looked at is using ACFTP with mode jumping. We do not expect this to present any problems as in this example chains should get closer together and therefore be easily coalesced. The mode jumping move is just a Metropolis-Hastings step (if somewhat elaborate), and we have seen from the examples of mixture distributions that inserting complicated moves at regular intervals in a standard chain does not stop chains coalescing. In fact it may aid the coalescing of chains because two chains whose current states are in the same mode will likely propose to jump to the same mode. The construction of the mode jumping move (both for Tjelmeland & Hegstad's (2001) method and the method described here), is probably going to result in the same proposal for both chains, and so they will coalesce if they both accept the proposed new state. In order to aid and achieve coalescence we also have the vector of allocation parameters δ. These will coalesce without any extra effort because they are discrete, and if two chains have the same value of δ then the other parameters may easily follow.

Appendix A

Obtaining the ACFTP Error Bound

To elaborate on the sequence of inequalities in (3.1) we let $\epsilon = \epsilon_T$ and $x = rs$ then the first two lines of (3.1) state that

$$(1-\epsilon)^x \epsilon \leq \left(1 - \frac{1}{x+1}\right)^{x+1} \frac{1}{x} .$$

This inequality is obtained by maximizing the left hand side over values of ϵ. Differentiating the left hand side with respect to ϵ gives

$$\frac{\mathrm{d}}{\mathrm{d}\epsilon}[(1-\epsilon)^x \epsilon] = (1-\epsilon)^x - x(1-\epsilon)^{x-1}\epsilon .$$

The maximum is obtained by setting this equal to 0 and solving for ϵ,

$$(1-\epsilon)^x - x(1-\epsilon)^{x-1}\epsilon = 0 ,$$

which is satisfied if $\epsilon = 1/(x+1)$. We can check that this indeed a maximum by differentiating again, substituting in $\epsilon = 1/(x+1)$ and simplifying to obtain,

$$\frac{\mathrm{d}}{\mathrm{d}\epsilon}[(1-\epsilon)^x - x(1-\epsilon)^{x-1}\epsilon](\epsilon = 1/(x+1)) = -\frac{\left(\frac{x}{x+1}\right)^x (x+1)^2}{x}$$

which is less than 0 for all $x > 0$.

Now using the results of the maximization we have that,

$$\begin{aligned}(1-\epsilon)^x\epsilon &\leq \left(1-\frac{1}{x+1}\right)^x \frac{1}{x+1} \\ &= \left(1-\frac{1}{x+1}\right)^{x+1} \frac{1}{x+1}\frac{x+1}{x} \\ &= \left(1-\frac{1}{x+1}\right)^{x+1} \frac{1}{x}\,, \end{aligned} \tag{A.1}$$

which confirms the inequality separating lines 1 and 2 of (3.1).
The third line in (3.1) is obtained by using the fact that,

$$\left(1-\frac{1}{y}\right)^y < \frac{1}{e} \text{ for all } y > 0\,.$$

Combining this fact with (A.1) gives that,

$$\left(1-\frac{1}{x+1}\right)^{x+1} \frac{1}{x} < \frac{1}{xe}$$

as stated in (3.1).

Appendix B

Neal's Coupling Proposal

The expression 4.1 for finding the nearest point to $x \in \mathbb{R}$ on the grid

$$G_x = \{\ldots, -\omega + 2\omega u_1,\ \omega + 2\omega u_1,\ 3\omega + 2\omega u_1, \ldots\}$$

is derived by considering $y = x - 2\omega u_1 + \omega$ and finding the nearest point to y on the transformed grid

$$G_y = \{\ldots, 0,\ 2\omega,\ 4\omega, \ldots\}.$$

The nearest point to y in G_y is

$$2\omega \left\lfloor \frac{y + \omega}{2\omega} \right\rfloor.$$

Transforming this back gives the nearest point to x in G_x as

$$2\omega \left\lfloor \frac{x - 2\omega u_1 + 2\omega}{2\omega} \right\rfloor + 2\omega u_1 - \omega = 2\omega \left\{ \left(u_1 - \frac{1}{2} \right) + \left\lfloor \frac{x}{2\omega} - u_1 + 1 \right\rfloor \right\}.$$

Appendix C

Multivariate Normal Covariance Matrices

1.000	0.331	0.008	-0.261	0.175	0.054
0.331	1.000	0.660	-0.143	-0.631	-0.002
0.008	0.660	1.000	-0.346	-0.446	0.426
-0.261	-0.143	-0.346	1.000	-0.581	-0.150
0.175	-0.631	-0.446	-0.581	1.000	-0.123
0.054	-0.002	0.426	-0.150	-0.123	1.000

Table C.1: Covariance matrix for MVN1.

1.000	-0.199	-0.199	-0.199	-0.199	-0.199
-0.199	1.000	-0.199	-0.199	-0.199	-0.199
-0.199	-0.199	1.000	-0.199	-0.199	-0.199
-0.199	-0.199	-0.199	1.000	-0.199	-0.199
-0.199	-0.199	-0.199	-0.199	1.000	-0.199
-0.199	-0.199	-0.199	-0.199	-0.199	1.000

Table C.2: Covariance matrix for MVN2.

1.000	-0.326	0.277	0.018	0.232	-0.064	-0.386	0.181	0.601	-0.241
-0.326	1.000	-0.097	-0.216	-0.088	0.085	0.452	0.182	-0.560	0.281
0.277	-0.097	1.000	0.109	0.014	0.057	-0.106	0.014	0.129	-0.042
0.018	-0.216	0.109	1.000	-0.085	-0.411	0.053	-0.025	-0.176	0.348
0.232	-0.088	0.014	-0.085	1.000	-0.041	-0.319	0.177	0.376	-0.021
-0.064	0.085	0.057	-0.411	-0.041	1.000	0.005	-0.080	-0.042	-0.209
-0.386	0.452	-0.106	0.053	-0.319	0.005	1.000	0.022	-0.928	0.049
0.181	0.182	0.014	-0.025	0.177	-0.080	0.022	1.000	0.083	0.142
0.601	-0.560	0.129	-0.176	0.376	-0.042	-0.928	0.083	1.000	-0.262
-0.241	0.281	-0.042	0.348	-0.021	-0.209	0.049	0.142	-0.262	1.000

Table C.3: Covariance matrix for MVN3.

First 20 rows									
1.00	0.11	-0.21	0.04	0.09	-0.26	-0.06	0.53	0.03	-0.18
0.11	1.00	-0.18	-0.13	0.49	0.25	-0.01	0.05	-0.08	0.32
-0.21	-0.18	1.00	0.18	0.10	-0.21	0.43	0.15	0.06	-0.14
0.04	-0.13	0.18	1.00	-0.49	0.17	0.11	0.21	-0.06	-0.14
0.09	0.49	0.10	-0.49	1.00	-0.03	0.17	0.14	0.02	-0.14
-0.26	0.25	-0.21	0.17	-0.03	1.00	-0.27	-0.50	0.15	-0.33
-0.06	-0.01	0.43	0.11	0.17	-0.27	1.00	0.11	0.01	0.17
0.53	0.05	0.15	0.21	0.14	-0.50	0.11	1.00	-0.09	-0.00
0.03	-0.08	0.06	-0.06	0.02	0.15	0.01	-0.09	1.00	-0.07
-0.18	0.32	-0.14	-0.14	-0.14	-0.33	0.17	-0.00	-0.07	1.00
0.22	0.04	-0.29	0.17	0.08	-0.23	-0.24	0.16	-0.12	0.06
-0.28	0.07	0.06	0.13	-0.03	0.02	0.10	0.22	-0.01	0.14
-0.11	-0.31	-0.08	0.22	-0.31	-0.07	0.22	-0.06	-0.05	-0.12
-0.61	-0.31	0.50	0.01	-0.08	-0.06	0.09	-0.11	0.04	0.21
0.06	-0.01	0.29	0.28	-0.04	0.11	-0.32	0.06	-0.27	-0.15
-0.03	-0.00	0.25	0.23	-0.08	-0.33	-0.09	0.23	-0.40	0.26
-0.14	-0.22	0.05	-0.26	0.15	0.02	-0.00	-0.01	-0.02	-0.07
0.35	0.17	-0.40	-0.14	0.02	-0.17	-0.08	0.29	0.07	0.27
-0.14	0.03	-0.14	0.15	-0.24	-0.09	0.14	0.03	-0.08	0.37
-0.01	-0.35	-0.09	-0.01	-0.20	0.16	-0.38	0.02	0.17	-0.51
Last 20 rows									
0.22	-0.28	-0.11	-0.61	0.06	-0.03	-0.14	0.35	-0.14	-0.01
0.04	0.07	-0.31	-0.31	-0.01	-0.00	-0.22	0.17	0.03	-0.35
-0.29	0.06	-0.08	0.50	0.29	0.25	0.05	-0.40	-0.14	-0.09
0.17	0.13	0.22	0.01	0.28	0.23	-0.26	-0.14	0.15	-0.01
0.08	-0.03	-0.31	-0.08	-0.04	-0.08	0.15	0.02	-0.24	-0.20
-0.23	0.02	-0.07	-0.06	0.11	-0.33	0.02	-0.17	-0.09	0.16
-0.24	0.10	0.22	0.09	-0.32	-0.09	-0.00	-0.08	0.14	-0.38
0.16	0.22	-0.06	-0.11	0.06	0.23	-0.01	0.29	0.03	0.02
-0.12	-0.01	-0.05	0.04	-0.27	-0.40	-0.02	0.07	-0.08	0.17
0.06	0.14	-0.12	0.21	-0.15	0.26	-0.07	0.27	0.37	-0.51
1.00	0.10	-0.08	-0.02	0.30	0.21	-0.21	-0.01	-0.04	-0.08
0.10	1.00	0.23	0.16	-0.06	0.08	-0.26	-0.01	-0.09	-0.09
-0.08	0.23	1.00	-0.30	-0.30	-0.22	-0.17	-0.23	-0.08	0.09
-0.02	0.16	-0.30	1.00	0.13	0.12	0.08	-0.34	0.17	-0.04
0.30	-0.06	-0.30	0.13	1.00	0.56	0.17	-0.37	-0.28	-0.01
0.21	0.08	-0.22	0.12	0.56	1.00	-0.04	0.05	0.15	-0.23
-0.21	-0.26	-0.17	0.08	0.17	-0.04	1.00	-0.08	-0.18	0.04
-0.01	-0.01	-0.23	-0.34	-0.37	0.05	-0.08	1.00	0.36	-0.12
-0.04	-0.09	-0.08	0.17	-0.28	0.15	-0.18	0.36	1.00	-0.16
-0.08	-0.09	0.09	-0.04	-0.01	-0.23	0.04	-0.12	-0.16	1.00

Table C.4: Covariance matrix for MVN4.

Appendix D

Further ACFTP Results for MVN Distributions

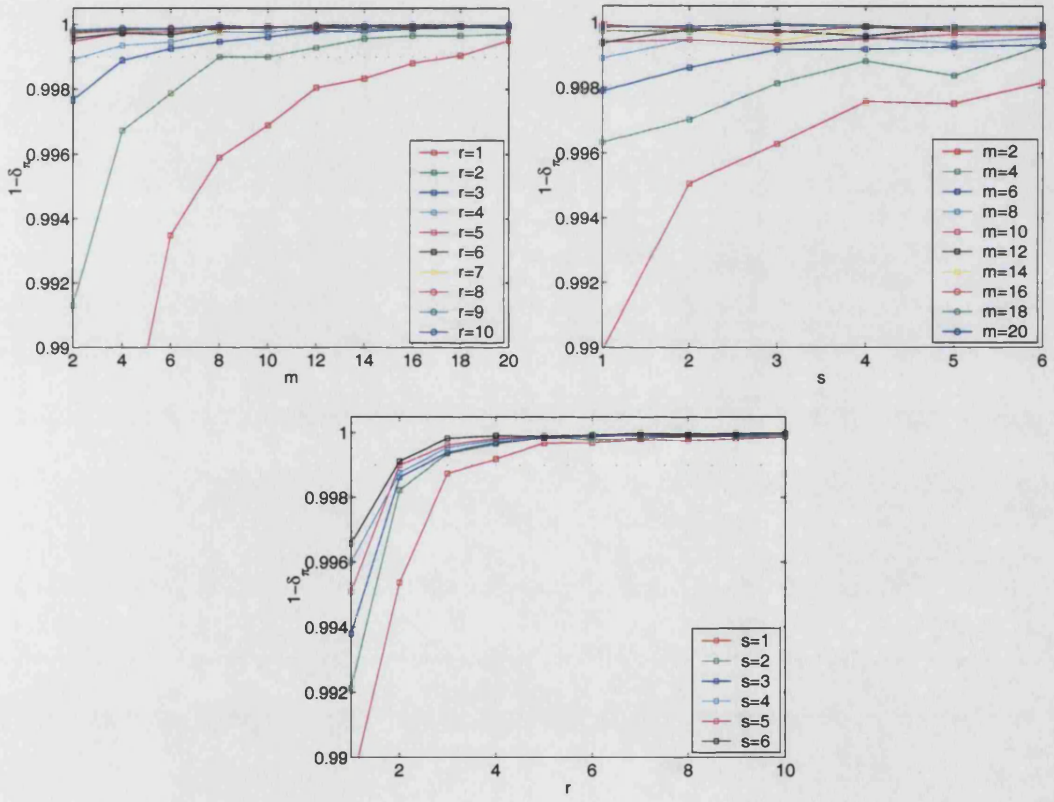

Figure D.1: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s for MVN2.

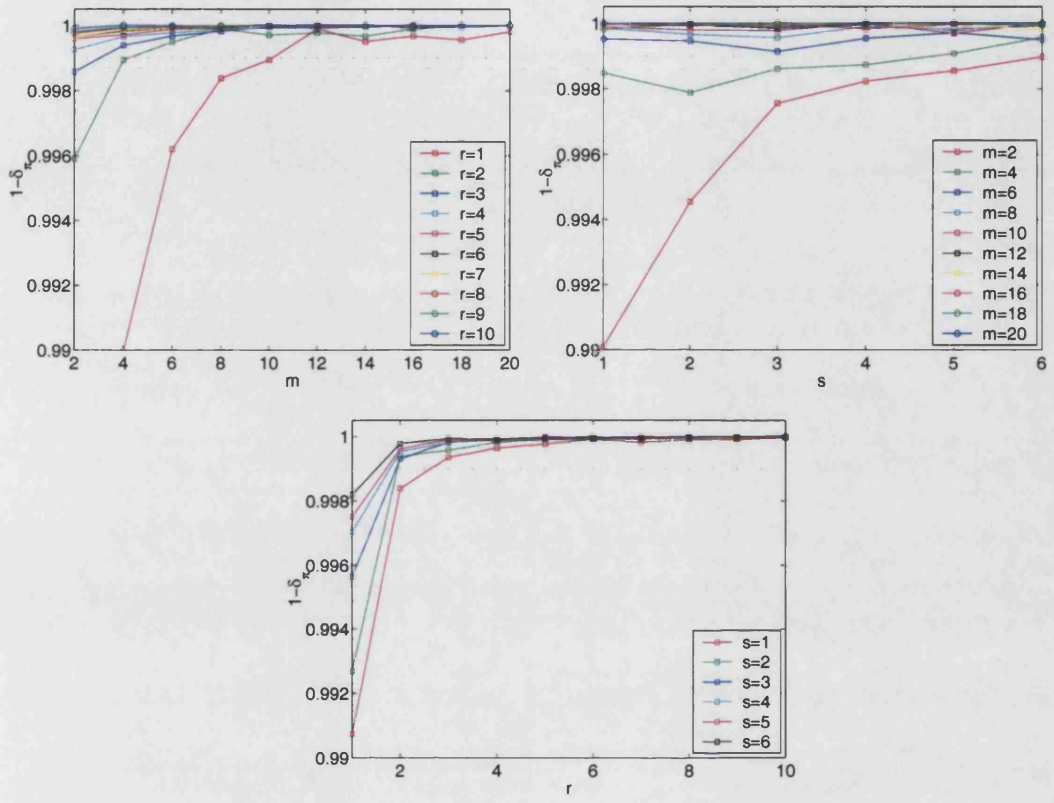

Figure D.2: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s for MVN3.

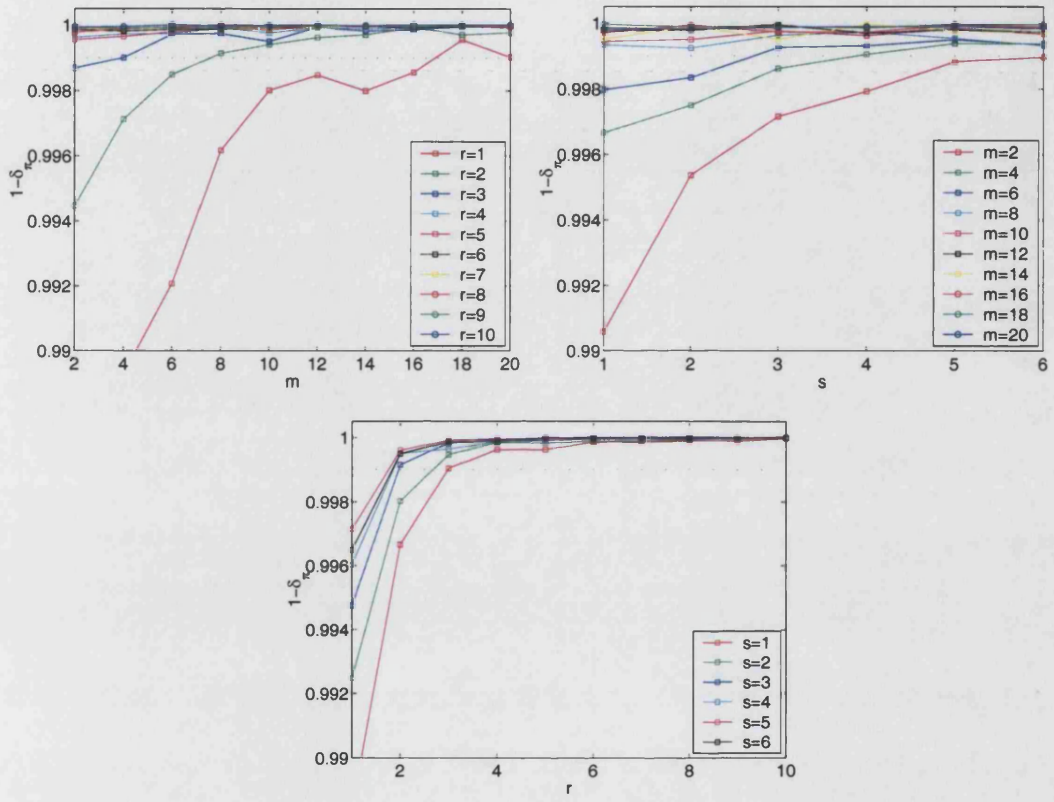

Figure D.3: Proportion of stationary chains coalescing with ACFTP chains for different combinations of m, r and s for MVN4.

Bibliography

Besag, J. & Green, P. J. (1993), 'Spatial Statistics and Bayesian Computation', *Journal of the Royal Statistical Society Series B* 55(1), 25–37.

Best, N. G., Cowles, M. K. & Vines, S. K. (1995), *CODA Manual version 0.30*, MRC Biostatistics Unit, Cambridge, UK.

Box, G. E. P. (1980), 'Sampling and Bayes Inference in Scientific Modelling and Robustness', *Journal of the Royal Statistical Society Series A* 143(4), 383–430.

Box, G. E. P. & Tiao, G. C. (1968), 'A Bayesian Approach to Some Outlier Problems', *Biometrika* 55, 119–129.

Breyer, L. A. & Roberts, G. O. (2000), 'Catalytic Perfect Simulation', *Methodology and Computing in Applied Probability: To appear* .

Brooks, S. P. (1997), 'Discussion to Richardson and Green (1997)'.

Brooks, S. P. & Gelman, A. (1998), 'General Methods for Monitoring Convergence of Iterative Simulations', *Journal of Computational and Graphical Statistics* 7(4), 434–455.

Brooks, S. P. & Giudici, P. (1999), Convergence Assessment for Reversible Jump MCMC Simulations, *in* J.M. Bernado et al., ed., 'Bayesian Statistics 6', Oxford University Press, Oxford, pp. 733–742.

Brooks, S. P. & Roberts, G. O. (1998), 'Convergence Assessment Techniques for Markov Chain Monte Carlo', *Statistics and Computing* 8, 319–335.

Carlin, B. P. & Louis, T. A. (1996), *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman and Hall, Chapter 6.

Casella, G., Lavine, M. & Robert, C. (2001), 'Explaining the Perfect Sampler', *The American Statistician* 55(4), 299–305.

Castelloe, J. M. & Zimmerman, D. L. (2002), Convergence Assessment for Reversible Jump MCMC Samplers, Technical Report 313, Department of Statistics and Actuarial Science, University of Iowa.

Childs, A. M., Patterson, R. B. & MacKay, D. J. C. (2001), 'Exact Sampling from Non-Attractive Distributions Using Summary States', *Physical Review E* 63(036113).

Corcoran, J. N. & Tweedie, R. L. (1998), Perfect Sampling from Independent Metropolis-Hastings Chains, Technical report, Department of Statistics, Colorado State University.

Cowles, M. K. & Carlin, B. P. (1996), 'Markov Chain Monte Carlo Convergence Diagnostics: A Comparative Review', *Journal of the American Statistical Association* 91(434), 883–904.

Damien, P., Wakefield, J. & Walker, S. (1999), 'Gibbs Sampling for Bayesian Non-Conjugate and Hierarchical Models by Using Auxiliary Variables', *Journal of the Royal Statistical Society Series B* 61, 331–344.

Damien, P. & Walker, S. (2001), 'Sampling Truncated Normal, Beta and Gamma Densities', *Journal of Computational and Graphical Statistics* 10, 206–215.

Devroye, L. (1986), *Non-Uniform Random Variate Generation*, Springer-Verlag, New York.

Edwards, R. G. & Sokal, A. D. (1988), 'Generalization of the Fortuin-Kasteleyn-Swendsen-Wang Representation and Monte-Carlo Algorithm', *Physical Review D* 38(6), 2009–2012.

Fill, J. A. (1997), 'An Interruptible Algorithm for Perfect Sampling via Markov Chains', *The Annals of Applied Probability* 8, 131–162.

Geisser, S. (1980), 'Discussion to Box (1980)'.

Gelfand, A. E., Smith, A. F. M. & Lee, T. (1992), 'Bayesian Analysis of Constrained Parameter and Truncated Data Problems Using Gibbs Sampling', *Journal of the American Statistical Association* 87(418), 523–532.

Gelman, A. & Rubin, D. B. (1992), 'Inference from Iterative Simulation Using Multiple Sequences', *Statistical Science* 7(4), 457–511.

Geman, S. & Geman, D. (1984), 'Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images', *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6(6), 721–741.

Geweke, J. (1992), Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments, *in* J.M. Bernado et al., ed., 'Bayesian Statistics 4', Oxford University Press, pp. 169–193.

Gilks, W. R. & Wild, P. (1992), 'Adaptive Rejection Sampling for Gibbs Sampling', *Applied Statistics* 41(2), 337–348.

Green, P. J. (1995), 'Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination', *Biometrika* 82(4), 711–732.

Green, P. J. (2003), Chapter 6: Trans-Dimensional Markov Chain Monte Carlo, *in* P. J. Green, N. L. Hjort & S. Richardson, eds, 'Highly Structured Stochastis Systems', Oxford University Press.

Green, P. J. & Murdoch, D. J. (1999), 'Exact Sampling for Bayesian Inference: Towards General Purpose Algorithms', *Bayesian Statistics* 6, 301–321.

Guglielmi, A., Holmes, C. C. & Walker, S. (2001), 'Perfect Simulation Involving a Continuous and Unbounded State Space', *Journal of Computational and Graphical Statistics : To appear* .

Guttman, I. (1973), 'Care and Handling of Univariate or Multivariate Outliers in Detecting Spuriosity - A Bayesian Approach', *Technometrics* 15(4), 723–738.

Häggström, O. & Nelander, K. (1998), 'Exact Sampling from Antimonotone Systems', *Statistica Neerlandica* (50), 360–380.

Hastings, W. K. (1970), 'Monte Carlo Sampling Methods Using Markov Chains and their Applications', *Biometrika* 57(1), 97–109.

Heidelberger, P. & Welch, P. D. (1983), 'Simulation Run Length Control in the Presence of an Initial Transient', *Operations Research* 31(6), 1109–1144.

Huber, M. (1998), Exact Sampling and Approximate Counting Techniques, *in* 'Proceedings of the 30th ACM Symposium on the Theory of Computing'.

Hurn, M. A., Justel, A. & Robert, C. P. (2003), 'Estimating Mixtures of Regressions', *Journal of Computational and Graphical Statistics* 12(1), 55–79.

Johnson, V. E. (1996), 'Studying Convergence of Markov Chain Monte Carlo Algorithms using Coupled Sample Paths', *Journal of the American Statistical Association, Theory and Methods* 91(433), 154–166.

Justel, A. & Peña, D. (1996), 'Gibbs Sampling will Fail in Outlier Problems with Strong Masking', *Journal of Computational and Graphical Statistics* 5(2), 176–189.

Justel, A. & Peña, D. (2001), 'Bayesian Unmasking in Linear Models', *Computational Statistics and Data Analysis* 36, 68–84.

Kendall, W. S. (1997), 'Perfect Simulation for the Area-Interaction Point Process', *Probability Perspective* pp. 218–234.

Kendall, W. S. & Møller, J. (2000), 'Perfect Simulation using Dominating Processes on Ordered Spaces, with Applications to Locally Stable Point Processes', *Advanced Applied Probability* 32, 844–865.

Meng, X. (2000), Towards a More General Propp-Wilson Algorithm: Multistage Backward Coupling, *in* 'Monte Carlo Methods', Vol. 26, American Mathematical Society, pp. 85–93.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. & Teller, E. (1953), 'Equation of State Calculations by Fast Computing Machines', *The Journal of Chemical Physics* 21(6), 1087–1092.

Mira, A., Møller, J. & Roberts, G. O. (2001), 'Perfect Slice Samplers', *Journal of the Royal Statistical Society Series B* 63(3), 593–606.

Mira, A. & Tierney, L. (2002), 'Efficiency and Convergence Properties of Slice Samplers', *Scandinavian Journal of Statistics* 29(1), 1–12.

Murdoch, D. J. & Green, P. J. (1998), 'Exact Sampling from a Continuous State Space', *Scandinavian Journal of Statistics* 29, 483–502.

Neal, R. M. (2002), Circularly-Coupled Markov Chain Sampling, Technical Report 9910, Department of Statistics, University of Toronto.

Nummelin, E. (1984), *General Irreducible Markov Chains and Non-negative Operators*, Cambridge University press.

Pettit, L. I. (1990), 'The Conditional Predictive Ordinate for the Normal Distribution', *Journal of the Royal Statistical Society Series B* 52(1), 175–184.

Philippe, A. & Robert, C. P. (2001), Perfect Simulation of Positive Gaussian Distributions, Technical Report IRMA 56, Laboratoire de Statistique et Probabilités, Universitè des Sciences et Tecnologies de Lille.

Propp, J. G. & Wilson, D. B. (1996), 'Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics', *Random Structures and Algorithms* 9, 223–252.

Raftery, A. E. (1997), Hypothesis Testing and Model Selection, *in* W.R. Gilks, S. Richardson and D.J. Spiegelhalter, ed., 'Markov Chain Monte Carlo in Practice', Chapman and Hall, Chapter 10, pp. 163–187.

Raftery, A. E. & Lewis, S. (1992*a*), How Many Iterations in the Gibbs Sampler, *in* J.M. Bernado et al., ed., 'Bayesian Statistics 4', Oxford University Press, pp. 763–773.

Raftery, A. E. & Lewis, S. M. (1992*b*), 'Comment: One Long Run with Diagnostics: Implementation Stategies for Markov Chain Monte Carlo', *Statistical Science* 7, 493–497.

Reutter, A. & Johnson, V. E. (1995), General Strategies for Assessing Convergence of MCMC Algorithms using Coupled Sample Paths, Technical report, Institute of Statistics and Decision Sciences, Duke University.

Richardson, S. & Green, P. J. (1997), 'On Bayesian Analysis of Mixtures with an Unknown Number of Components', *Journal of the Royal Statistical Society Series B* 59(4), 731–792.

Roberts, G. O. & Rosenthal, J. S. (1999*a*), 'Convergence of Slice Sampler Markov Chains', *Journal of the Royal Statistical Society Series B* 61, 643–660.

Roberts, G. O. & Rosenthal, J. S. (1999*b*), 'The Polar Slice Sampler', *Stochastic Models* 18(2), 257–280.

Rousseeuw, P. J. (1984), 'Least Median Squares Regression', *Journal of the American Statistical Association* 79(388), 871–880.

Smith, B. J. (2001), *Bayesian Output Analysis Program (BOA) Version 1.0.0 Users Manual.*

Spiegelhalter, D. J., Thomas, A., Best, N. G. & Gilks, W. R. (1996), BUGS Examples Volume 1, Version 0.5, (version ii).

Stephens, M. (2000), 'Bayesian Analysis of Mixture Models with an Unknown Number of Components - An Alternative to Reversible Jump Methods', *Annals of Statistics* 28(1), 40–74.

Tierney, L. (1994), 'Markov Chains for Exploring Posterior Distributions', *Annals of Statistics* 22, 1701–1762.

Tjelmeland, H. & Hegstad, B. K. (2001), 'Mode Jumping Proposals in MCMC', *Scandinavian Journal of Statistics* 28(1), 205–223.

Verdinelli, I. & Wasserman, L. (1991), 'Bayesian Analysis of Outlier Problems using the Gibbs Sampler', *Statistics and Computing* 1, 105–117.

Wilson, D. B. (2000*a*), 'How to Couple from the Past Using a Read-Once Source of Randomness', *Random Structures and Algorithms* 16(1), 85–113.

Wilson, D. B. (2000*b*), 'Layered Multishift Coupling for use in Perfect Sampling Algorithms (with a primer on CFTP)', *Monte Carlo Methods, Fields Institute Communications, American Mathematical Society* 26, 141–176.